UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE CIENCIAS MATEMÁTICAS

Departamento de Estadística e Investigación Operativa I

TESIS DOCTORAL

Segmentación jerárquica en redes : aplicaciones

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Edwin Zarrazola Rivera

Directores

Javier Yáñez Gestoso
Daniel Gómez Gónzález

Madrid, 2014

SEGMENTACIÓN JERÁRQUICA EN REDES. APLICACIONES

Memoria presentada para optar al título de Doctor en Métodos Estadístico Matemáticos y Computacionales para el Tratamiento de la Información

Por:

EDWIN ZARRAZOLA RIVERA

Dirigida por:

Dr. Javier Yáñez Gestoso,
&
Dr. Daniel Gómez González

Facultad de Ciencias Matemáticas
Departamento de Estadística e Investigación Operativa I

UNIVERSIDAD COMPLUTENSE DE MADRID

2013

Resumen

La técnica que se propone a continuación se clasifica dentro de los procesos no supervisados de partición de redes a través de grafos, y tratará de dar una solución al problema de dividir una red a través de un enfoque jerárquico, utilizando como base la construcción de bosques soporte asociados al grafo de la red en estudio, exigiendo que estas estructuras cumplan algunas condiciones para lograr una "buena" partición jerárquica de la red.

Abstract

The technique proposed here is classified as unsupervised network partition processes through graphs, and try to give a solution to the problem of dividing a network via a hierarchical approach, using as a basis the construction of spanning forest associated to the graph, and demanding some conditions to these structures in order to achieve a "good" hierarchical network partition.

Prólogo

En esta memoria de tesis tratamos uno de los problemas propios en redes, como es el problema de la partición jerárquica de la red (denotado por HCNP) a través del grafo asociado a la red. En este sentido, en este trabajo se propone un algoritmo no supervisado, que hemos llamado Algoritmo Divide-and-Link (denotado D&L); el cual permite obtener una partición jerárquica de una red con un coste computacional bajo (el algoritmo D&L tiene una complejidad polinomial) que incluso le permite ser utilizado en redes grandes.

La inspiración para realizar este trabajo surge a partir de algunos trabajos previos realizados desde el año 2006 por los profesores de la Universidad Complutense de Madrid: Javier Yánez, Daniel Gómez y Javier Montero; los cuales plantearon las bases de lo que ahora presentamos como el Algoritmo Divide-and-Link. Sin embargo, el estudio del problema de partición en redes proviene de mucho tiempo atrás (como se muestra en el Capítulo 1), y la gran mayoría de los procedimientos que se encuentran en la literatura, lo que buscan es entregar una "buena" partición de la red, siguiendo algún criterio de calidad. No obstante, en este trabajo lo que buscamos es entregar una "buena secuencia" de particiones de la red, con la condición de que dicha secuencia sea jerárquica, de acuerdo a la definición de jerarquía se detalla en el Capítulo 2.

Algo interesante de señalar en el proceso que realiza el algoritmo Divide-and-Link que se propone en este trabajo, es que permite ser utilizado en diversas aplicaciones como se detalla a continuación.

La detección de comunidades en redes sociales (Capítulo 3), donde nuestro algoritmo utiliza la información de la red a través de medidas estructurales de disimilitud y afinidad asociadas a los enlaces (o aristas) de la red, para entregar como "salida" la forma dinámica de cómo se forman las comunidades en la red, de forma divisiva. Esta información está contenida en la partición jerárquica de la red y en el dendograma asociado a dicha jerarquización. Al final del Capítulo 3 se muestra la calidad de la partición realizada por el algoritmo Divide-and-Link al ser comparado con algunos de los algoritmos más relevantes en la literatura y que permiten realizar particiones jerárquicas.

En segmentación jerárquica de imágenes, desarrollado a lo largo del Capítulo 4, el algoritmo Divide-and-Link ha mostrado ser muy útil debido a que su forma jerárquica de trabajar permite ver la "película" de partición de la imagen, y así el investigador tiene la opción de escoger la parte de la secuencia que le sea más informativa; y no tiene la limitación propia de muchos algoritmos de segmentación de imágenes que entregan una sola partición de la imagen creada a partir de algún criterio y que en ocasiones, esa única

"salida" no es informativa. Por otro lado, el algoritmo Divide-and-Link utiliza toda la información de las imágenes a color, ya sea a través de la utilización de espacios de color aproximadamente uniformes, o utilizando medidas de color en ambiente borroso, y no se limita a llevar las imágenes a una escala de intensidad en una dimensión (pérdida de información), como así lo hacen algunos procedimientos de segmentación de imágenes muy utilizados por la comunidad científica (por ejemplo, algunos procedimientos que trae incorporados el Matlab).

En astronomía, en particular, en el problema de detección de regiones débiles en imágenes astronómicas (Subsecciones 4.8 – 4.9); el procedimiento que utiliza el algoritmo Divide-and-Link ha mostrado ser de bastante utilidad, pues la gran mayoría de los procedimientos de segmentación de imágenes concentran sus esfuerzos en las regiones más intensas o en el dual, es decir, lo que suele llamarse como "el fondo"; sin embargo, la identificación de las regiones débiles en el espacio es un problema diferente, raramente encontrado en la bibliografía especializada. Este problema es uno de los objetivos de la misión espacial Word Space Observatory Ultraviolet (WSO-UV), y en la cual participamos construyendo un algoritmo, a partir del Divide-and-Link, que permita identificar de manera automática las regiones débiles en imágenes astronómicas grandes, o en un conjunto de imágenes correspondientes a una región del espacio.

El alcance del algoritmo Divide-and-Link puede ser extendido a otros problemas como, por ejemplo, el análisis de imágenes hiperespectrales, las cuales ofrecen mucha más información en cada pixel, y en las cuales se espera que los algoritmos de segmentación utilicen toda la información disponible en la imagen. También, el procedimiento usado por nuestro algoritmo admite la utilización de grafos borrosos, en donde se debe tener en cuenta la forma de realizar comparación de los números borrosos asociados a las aristas (ver Capítulo 5).

Al final de la memoria se incluye un Apéndice A, que contiene dos formas de programar (en Matlab) el algoritmo D&L para el caso de imágenes. Una de ellas es matricial, en donde se tiene en cuenta la simetría de las redes asociadas a la imagen, y la otra forma es vectorial, a través de punteros se mueve en la red. La segunda forma de programación se asemeja a las técnicas que se utilizan en elementos finitos para dominios generales para resolver problemas de ecuaciones diferenciales parciales.

A partir del Capítulo 2, todas las aportaciones son originales; en concreto, los algoritmos D&L y D&LF, y las adaptaciones de estos algoritmos a los diversos problemas estudiados en esta memoria: detección de comunidades en redes sociales (Capítulo 3), segmentación jerárquica de imágenes digitales (Capítulo 4), detección de regiones débiles en imágenes astronómicas (Capítulo 4), así como las mejoras computacionales (Capítulo 4).

Finalmente, la realización de este trabajo no hubiera sido posible sin la ayuda de los profesores Javier Yáñez y Daniel Gómez que siempre han tenido la agudeza necesaria para resolver todas las cuestiones académicas que nos hemos encontrado a lo largo del desarrollo de este trabajo. También quiero agradecer a mi familia, que desde la distancia siempre me han apoyado en todo, y a todos mis amigos que siempre fueron una voz de aliento.

Edwin Zarrazola R.

Índice general

Capítulo 1

Introducción

Contenido

La agrupación de objetos similares para producir una "clasificación", es una de las habilidades más básicas (y primitivas) que poseen en general los seres vivos. Es claro que para la supervivencia, muchas especies deben darse cuenta que muchos individuos del entorno comparten ciertas propiedades, como ser comestible, o venenoso o feroz, entre otras. En la biología, la clasificación ha sido una preocupación desde las primeras investigaciones Biológicas. Aristóteles construyó un elaborado sistema de clasificación de las especies del reino animal, el cual comenzó dividiendo los animales en dos grupos principales, los que tienen sangre roja (que corresponde aproximadamente a nuestros vertebrados) y los que carecen de ella (los invertebrados). Luego, subdivide estos dos grupos de acuerdo a la forma en que se nacen, sea a través de huevos o no, como pupas (larvas) y así sucesivamente. Siguiendo a Aristóteles, Teofrasto de Ereso escribió los primeros relatos fundamentales de la estructura y clasificación de las plantas.

La clasificación de los animales y las plantas ha jugado claramente un papel importante en los campos de la biología y la zoología, sobre todo como base para la teoría

de la evolución de Darwin. Sin embargo, "la clasificación" también ha jugado un papel central en el desarrollo de muchos otros campos de la ciencia. Así por ejemplo, la clasificación de los elementos en la tabla periódica producida por Mendeleyev en la década de 1860, ha tenido un profundo impacto en la comprensión de la estructura del átomo. En astronomía, la clasificación de estrellas en enanas y gigantes utilizando el diagrama de Hertzsprung-Russell de temperatura frente intensidad ha afectado fuertemente las teorías de la evolución estelar.

En general, un sistema de clasificación puede representar simplemente un método conveniente para la organización de un gran conjunto de datos de manera que pueda ser entendido más fácilmente y recuperar la información de forma más eficiente. Si los datos se pueden resumir, de manera válida, por un pequeño número de grupos de objetos, entonces las etiquetas de cada grupo pueden proporcionar una descripción muy concisa de los patrones de similitudes y diferencias en los datos. En la investigación de mercados, por ejemplo, estas clasificaciones suelen ser útiles para agrupar un gran número de los encuestados de acuerdo con sus preferencias respecto a ciertos productos y ayudar a identificar, entre otras cosas, preferencias de ciertos productos para un determinado tipo de consumidor.

La necesidad de explicar conjuntos de datos de esta manera es cada vez más importante debido al creciente número de grandes bases de datos disponibles en la actualidad en las diferentes áreas del saber. La exploración de esas bases de datos mediante el análisis de conglomerados y otras técnicas de análisis multivariante suele llamarse "minería de datos".

1.1. Dividir y/o Agrupar

El "análisis de agrupamiento" aparece frecuentemente para resolver cuestiones en disciplinas como biología, botánica, medicina, psicología, geografía, marketing, procesamiento de imágenes, psiquiatría, arqueología, entre otras.

Lo que se espera de los diferentes métodos numéricos relacionados con agrupación es que las agrupaciones sean objetivas y estables. Que los métodos sean "objetivos" se refieren a que el análisis del mismo conjunto de individuos por los mismos métodos numéricos produzcan la misma clasificación; y "estable" en el sentido que la clasificación permanece igual cuando se adicionan individuos o nuevas características que los describan.

Una gama amplia de nombres se han utilizado para estos métodos numéricos dependiendo en gran medida del área de aplicación. Taxonomía numérica se utiliza generalmente en la biología. En psicología, el término "análisis Q" es empleado con frecuencia. En inteligencia artificial el nombre favorito suele ser "reconocimiento de patrones". En investigación de mercados y tratamiento de imágenes se suele hablar de "segmentación". Pero hoy en día el "análisis cluster" es probablemente el término genérico que más se utiliza para los procedimientos que tratan de descubrir los grupos de datos. En la mayoría de aplicaciones se busca una partición de los datos, en el que cada individuo u objeto pertenece a un solo grupo, y el conjunto completo de grupos contiene todos los individuos. En algunas circunstancias, sin embargo, las agrupaciones superpuestas pueden proporcionar una solución más aceptable.

A continuación se describe brevemente una serie de aplicaciones de análisis de conglomerados en algunas de estas disciplinas (véase, por ejemplo, [59, 4, 87, 25]).

Estudios de mercado

Una de las estrategias básicas del estudio de mercados es dividir los clientes en grupos homogéneos. Así, por ejemplo, pueden identificar los grupos de consumidores que buscan beneficios similares de un producto. Con esta información la empresa puede implementar estrategias que optimicen su comunicación con cada grupo de clientes.

Otro ejemplo en esta disciplina es la de los analistas de mercado, pues uno de sus intereses es la agrupación de características financieras de las empresas con el fin de ser capaz de relacionarlos con su evolución bursátil. Información adicional de estos ejemplos pueden verse en [59, 18].

Astronomía

El análisis Cluster se puede utilizar para clasificar los objetos astronómicos, y a menudo puede ayudar a los astrónomos a encontrar objetos extraños dentro de un flujo de datos. Algunos ejemplos son los descubrimientos de dos tipos de cuásares de alto corrimiento al rojo (muy luminosos, núcleos galácticos activos, cuyos centros están oscurecidas por polvo y gas), y las enanas marrones.

Un ejemplo específico es el estudio publicado en 1996 por Faúndez-Abans et al. (ver [35]), el cual aplica una técnica de agrupación propuesta por Ward (ver [105]) a los datos sobre la composición química de 192 nebulosas planetarias. Se identificaron seis grupos que eran similares en muchos aspectos a una clasificación usada previamente de tales objetos, pero que también mostró diferencias interesantes.

Un segundo ejemplo astronómico es propuesto por Celeux y Govaert en [17], en el cual aplican los modelos normales mixtos a los datos estelares que consisten en una población de 2.370 estrellas descritas por sus velocidades hacia el centro galáctico y hacia la rotación galáctica. La investigación proporciona un modelo de tres grupos, uno de tamaño grande y volumen pequeño, y otros dos de tamaño pequeño y volumen grande. Una exposición más amplia de la utilización de análisis de conglomerados en astronomía es expuesta en [4].

Psiquiatría

En general, las enfermedades mentales suelen ser más difíciles de diagnosticar que las físicas, y en psiquiatría ha habido mucho interés en el uso de técnicas de análisis de conglomerados para refinar o incluso redefinir las categorías de diagnóstico actuales.

Gran parte de este trabajo en esta área se ha concentrado en el análisis de la depresión en pacientes, en particular; hay un cierto interés en la cuestión de la existencia de subtipos endógenos y neurótico. En 1969, Pilowsky et al. (ver [87]), usaron un método de agrupación descrito en [101], para agrupar 200 pacientes basados en sus respuestas a un cuestionario para medir el grado de depresión, junto con la información sobre su estado mental, el sexo, la edad y la duración de la enfermedad (observe una vez más los diferentes tipos de variables involucradas). Uno de los grupos producidos fue identificado con la depresión endógena. Un estudio similar realizado por Paykel (ver [84]), con 165 pacientes (y utilizando un método de agrupación publicado en [40]), encontró cuatro grupos, uno de los cuales claramente era el de depresión psicótica.

Bioinformática y genética

En los últimos años es claro el enorme crecimiento de la Bioinformática, que es la unión de la biología molecular, la informática, las matemáticas y la estadística. Las técnicas de biología molecular han dejado claro que los grandes eventos en la vida de una célula están regulados por factores que alteran la expresión de los genes. Una actividad importante en la biología moderna es intentar comprender cómo se controla selectivamente la expresión de los genes. Las micromatrices de ADN (ver Cortese, [25]) son un avance revolucionario en la biología molecular experimental que tiene la capacidad de estudiar simultáneamente miles de genes bajo una multitud de condiciones y proporcionar un volumen grande de información para el investigador. Estos nuevos tipos de datos comparten una característica común: el número de variables "p" excede en gran medida el número de observaciones "n"; es decir, alta dimensional. Muchos métodos estadísticos clásicos no se pueden aplicar a los datos de alta dimensión sin tener que realizar modificaciones sustanciales. Sin embargo, el análisis de conglomerados se puede utilizar para identificar grupos de genes con patrones similares de expresión, y esto puede ayudar a proporcionar respuestas a las preguntas de cómo la expresión del gen se ve afectada por diversas enfermedades y cuáles genes son responsables de ciertas enfermedades hereditarias. Selinski y Ickstadt (ver [94]), utilizan análisis de conglomerados en los polimorfismos de nucleótido simple para detectar diferencias entre los individuos enfermos y otros de control en estudios de casos-control; y Eisen et al. en [29] utilizan la agrupación de datos con expresiones amplias del genoma para identificar subtipos de cáncer asociados con la supervivencia; En 2010, Witten y Tibshirani (ver [109]) describen una aplicación similar al agrupar los datos de carcinoma de células renales.

Otras aplicaciones en áreas como las redes sociales y el tratamiento de imágenes digitales, serán explicadas con más detalle en la Sección 1.3 y la Sección 1.4, respectivamente.

1.2. Segmentación jerárquica

Cuando se habla de clasificación jerárquica de los datos, éstos no se dividen en un número particular de las clases o grupos en un solo paso. En lugar ello, la clasificación jerárquica va a consistir en una serie de particiones, que pueden ejecutarse desde un solo grupo que contiene todos los individuos, hasta n grupos que contienen cada uno un solo individuo. Las técnicas de segmentación jerárquica suelen catalogarse en dos clases: los **métodos aglomerativos**, que realizan de una serie de fusiones sucesivas de los n individuos en grupos, y los **métodos divisivos**, que separan los n individuos sucesivamente en agrupaciones más finas. Muchas de estas técnicas de agrupación jerárquica buscan encontrar el paso óptimo, según algún criterio, en cada etapa en la subdivisión progresiva o síntesis de los datos. Las agrupaciones jerárquicas producidos por métodos aglomerativos o divisivos, pueden ser representados por un diagrama de dos dimensiones conocido como un dendograma, que ilustra las fusiones o divisiones hechas en cada etapa del análisis.

1.2.1. La mejor partición: problema del número de grupos

En ocasiones puede ocurrir que un investigador no esté interesado en el proceso jerárquico completo, sino en una o dos particiones obtenidas a partir de ella, y esto implica decidir sobre el número de grupos presentes. Hay una variedad de métodos formales que se aplican igualmente bien a la agrupación jerárquica y métodos de optimización, sin embargo, nos centramos en métodos específicos para técnicas jerárquicas.

En agrupaciones aglomerativas o divisivas, las particiones se logran mediante la selección de una de las soluciones en la secuencia anidada de agrupamientos que componen la jerarquía, lo que equivale al corte de un dendograma a una altura particular, (a veces denominado el mejor corte). Esto define una partición de tal manera que las agrupaciones por debajo de dicha altura son distantes entre sí por lo menos esa cantidad, y el dendograma puede sugerir el número de grupos. Para indicar el mejor corte se suelen buscar grandes cambios en los niveles de agrupación. Un desarrollo más flexible de esta idea, publicada en 2008 por Langfelder et al., (ver [69]), es "el corte dinámico de árboles" el cual permite que las diferentes ramas del árbol sean cortadas en diferentes niveles. Esto se ha implementado como un paquete en R (`dynamic-TreeCut`), y es particularmente apropiado cuando hay conjuntos de grupos anidados.

Enfoques adicionales relacionados con el problema de determinar el número de grupos, se pueden estudiar en Milligan y Cooper [74], Gordon [53], Duda y Hart [28] y Beale [5], en función del problema que se aborda.

1.3. Redes sociales

Las redes sociales suelen representar las relaciones entre "entidades" sociales; ya sean relaciones dadas por las comunicaciones entre los miembros de un grupo o de las relaciones económicas entre las empresas, entre otras. Muchos análisis de redes sociales se centran en el análisis estructural de las redes, todo esto con el fin de ayudar a explicar el comportamiento social. Estos métodos son tradicionalmente usados en las ciencias sociales y del comportamiento, pero hoy en día se aplican también en economía, marketing y otras áreas también. En esta sección, se presentan algunos tipos de redes sociales así como algunos métodos bien conocidos para el análisis de redes sociales.

1.3.1. Tipos de Redes Sociales

Las redes sociales se clasifican dependiendo de la naturaleza del conjunto de individuos y las propiedades de las relaciones entre ellos (véase, por ejemplo, [106, 93, 39, 107]). Los individuos o entidades pueden ser de diversos tipos, tales como personas, organizaciones o una colección o conjunto de personas u organizaciones. Un grupo de personas podría ser, por ejemplo, un grupo de estudiantes que asisten a la misma conferencia y un conjunto de organizaciones podría ser, por ejemplo, un conjunto de comunidades de un país. Las relaciones observadas a nivel de parejas de individuos pueden ser de diversos tipos, por ejemplo, grado de amistad (ver [106]), transferencia de bienes (tangibles o intangibles), interacciones físicas, entre muchas otras.

Las relaciones en una red se establecen a través de las relaciones entre sus individuos o "actores". Estas pueden ser, por ejemplo, relaciones de poder dentro de una organización, de amistad o confianza entre las personas, de colaboración, contractuales,

etc. Muchas de estas relaciones son difíciles de medir, ya que por lo general no son explícitas o sólo una parte de dicho conocimiento es asequible (ver, por ejemplo, [106]). Las redes de interacción son un tipo de redes en las cuales las relaciones se dan por las diferentes interacciones entre dos de individuos. Así, un contacto (por ejemplo, por correo electrónico o por teléfono) o una transmisión (por ejemplo, una enfermedad) hacen parte de este tipo de redes. Una forma particular de las redes de interacción suelen llamarse "redes de transmisión" en donde se transmiten entre los nodos materiales tangibles (como gas, agua, etc) o intangibles (como dinero, información, etc) y que generalmente es medible.

1.3.2. Análisis de redes sociales

En las ciencias sociales, el análisis de redes tiene una larga tradición (ver [39]). El objetivo principal del análisis de redes sociales es detectar e interpretar los patrones de las relaciones sociales entre los individuos. Sin embargo, recientemente, el campo también ganó popularidad en muchas áreas de investigación como la inmunología, los sistemas de transporte, la biología molecular, los sistemas de información, sistemas de computación y la ciencia organizacional. A pesar de que el dominio de la aplicación determina la forma apropiada de análisis, los métodos que son frecuentes en el análisis de redes se pueden distinguir por el nivel de análisis. Brandes y Erlebach en [10] proponen tres niveles de análisis:

- **Análisis a nivel de elementos**. Una de las cuestiones fundamentales (entre muchas otras) en este tipo de análisis es detectar la relevancia de un elemento de red, es decir, de un vértice o una arista: ¿Qué tan importante es este individuo (nodo) en la red?. Habitualmente, las medidas para evaluar la "importancia" de un individuo se basan en su "centralidad" en la red. La centralidad de los elementos de las redes sociales se pueden determinar utilizando diferentes estadísticas locales, tales como el grado (entrada/salida), el coeficiente de agrupación para evaluar la importancia de un nodo, la intermediación (betweenness) para evaluar la importancia de una arista de la red, entre muchas otras. Ejemplos de aplicación de este tipo de análisis pueden encontrarse en [9, 67].

- **Análisis a nivel de grupos**. Un objetivo común en las redes sociales es separar en grupos a los individuos que tiene conexiones (también llamadas aristas o bordes) fuertes. Por ejemplo, en las redes de interacción, las personas que forman un grupo en particular interactúan más estrechamente entre sí que con individuos de otros grupos de la red. Un grupo se caracteriza (entre otras cosas) por las fuertes relaciones (conexiones) entre sus miembros. El subgrafo inducido por este grupo tiene una mayor conectividad entre cada par de miembros dentro del grupo en comparación con nodos fuera del grupo. El punto de partida de todos estos conceptos se relaciona con la idea de subgrafos cohesivos (ver [106, 93]).

- **Análisis a nivel de redes**. Este enfoque pretende estudiar las propiedades de la red en su conjunto. Las propiedades de red son señaladas para evaluar la similitud entre redes, además, los estadísticos obtenidos en la red muchas veces reflejan rasgos característicos con otras redes, por ejemplo, en un cierto dominio de aplicación. Por otra parte, observar cambios en las propiedades de una red

podría proporcionar indicaciones importantes para la interpretación del análisis temporal (dinámico) de la red. Para la descripción de una red existen algunos estadísticos globales tales como el diámetro, la densidad, el coeficiente de agrupación del grafo, la distancia media geodésica, entre otras.

1.3.3. Segmentación en Redes

El problema de clustering generalmente busca disponer una familia de ítems en grupos homogéneos, teniendo en cuenta información cualitativa y cuantitativa inherente en ellos. Sin embargo, la implementación práctica de algunas técnicas de clustering requieren ciertos supuestos matemáticos que algunas veces son difíciles, sino imposibles, de cumplir. Algunos de estos supuestos, como por ejemplo la independencia estadística, suele ser una condición estándar exigida para poder aplicar varios modelos teóricos.

Alrededor de los modelos de clustering (o agrupación) aún se encuentra mucho trabajo por delante, por ejemplo en la manera de identificar cómo objetos reales son conectados (véase por ejemplo, [2] para un ejemplo de un modelo de agrupación sin estructura, y la modificación propuesta en [75]). De hecho, muchos conjuntos de datos pueden estructurarse por medio de un grafo, es decir, un conjunto de nodos unidos por aristas. Si las aristas tienen un peso (o valor) entonces lo que se obtiene es una red.

1.3.4. El Problema de Detección de Comunidades

En esta parte, nos centraremos en los diferentes algoritmos que se han propuesto para los problemas de detección de la comunidades. Un gran número de algoritmos se han propuesto recientemente para la identificación de comunidades (para más información, en Fortunato [37], se hace una revisión bastante exhaustiva). Una clasificación simple basada en los resultados entregados por los algoritmos de detección de comunidades, nos permite una división (como sucede en los problemas clásicos de clustering) de los algoritmos de segmentación entre jerárquicos y no jerárquicos. Como es señalado en [37, 80], la mayoría de los algoritmos que se ocupan de los problemas de detección de comunidades se han desarrollado para obtener un partición no jerárquica del grafo. Sin embargo, algunos de ellos se pueden adaptar para dar una agrupación jerárquica de la red. Ahora le damos una breve reseña de algunos de ellos distinguidos en cuatro tipos o clases de algoritmos:

Algoritmos basados en matrices de disimilaridad

Estas técnicas se basan en el cálculo de disimilaridades (o similaridades) W_{ij} para cada par de nodos (i, j) (para más detalles, ver [37, 64, 1, 32, 33, 38, 92, 114, 77]). Estos pesos representan qué tan distintos (o parecidos) son cada par de nodos. Una vez que esta matriz de disimilitud o distancia W se ha construido, los nodos de la red se pueden agrupar usando cualquier técnica de agrupación jerárquica sin tener en cuenta la estructura. Por ejemplo, los nodos en el grafo pueden ser agrupados jerárquicamente comenzando con el par de nodos con el peso W_{ij} más fuerte y continuando con el proceso hasta llegar a los de peso más débil (en [62] puede consultarse una revisión de estas diferentes técnicas jerárquicas clásicas). En este grupo también podemos clasificar a los algoritmos espectrales que obtienen una segmentación jerárquica de la red, los cuales transforman los nodos del grafo en puntos de R^g usando g auto-vectores de la matriz

adyacente, entre otros (ver, por ejemplo, a Donetti y Muñoz [26] para más detalles). Nosotros hemos considerado, en el capítulo de detección de redes, un ejemplo de esta clase de algoritmos: se ha tomado el trabajo reciente de Newman en 2012 (ver [77] para más detalles). A partir de ahora, vamos a denotarlo como algoritmo **N2012**; y que consiste en los siguientes dos pasos:

1. Para cada par de nodos (no necesariamente adjacentes) $i, j \in V$ calcular el coeficiente de Jaccard
$$\sigma_{i,j} = \frac{n_{ij}}{(k_i k_j)^{1/2}}$$
donde n_{ij} es el número de vecinos comunes y k_i es el grado del nodo i. Esta medida toma valores en el intervalo $[0, 1]$.

2. Aplicar el algoritmo clásico aglomerativo (*average linkage clustering*) para obtener una segmentación jerárquica de la red.

Es evidente que tales técnicas clásicas pueden presentar inconvenientes mientras no se considere la estructura subyacente. Algunos de los principales problemas son descritos en [43, 80] (inconvenientes como memoria requerida y supuestos necesarios para construir la matriz W, entre otros).

Algoritmos divisivos

El trabajo pionero es debido a Girvan y Newman [43], y en el cual no se tiene en cuenta la construcción de una matriz de disimilitud (o afinidad) para todos los pares de nodos. Girvan y Newman presentaron un algoritmo de división (que denotamos como algoritmo-GN) con un buen desempeño en redes pequeñas y medianas. El algoritmo se basa en el cálculo de las ponderaciones w_{ij} para cada par de nodos adyacentes en la red (y no para todo par de nodos, como en los métodos tradicionales). El peso w_{ij} para un enlace representa su poder de "intermediación" en la estructura de comunicación definida por la red. Una vez que se define el peso, el algoritmo de división se puede resumir de la siguiente manera:

1. Calcular los pesos w_{ij} para todas las aristas de la red.

2. Eliminar las aristas con mayor w_{ij}.

3. Recalcular w_{ij} para todas las aristas afectadas por la eliminación.

4. Repetir desde el paso 2 hasta que no queden aristas.

Este algoritmo proporciona una segmentación jerárquica de los nodos del grafo y los resultados dependerán de cómo se han definido los pesos w_{ij}. Alternativamente, Girvan y Newman [43] introdujeron para el caso no-ponderado (sin pesos) el concepto de "betweenness", la cual puede entenderse como la frecuencia en que una arista participa en la comunicación de los diferentes pares de nodos. Esta frecuencia se puede obtener usando dos definiciones alternativas: la *intermediación en camino menor* (shortest path (SP) betweenness), en la que sólo se consideran caminos geodésicos; y la *intermediación en paseo aleatorio* (random-walk betweenness), definida por Newman en [79]. En la literatura es posible encontrar algunos métodos para la elección de un peso adecuado

w_{ij} junto con pequeñas modificaciones a este algoritmo divisivo (ver, por ejemplo, [117, 89, 78]). Siguiendo la filosofía del algoritmo GN, Radicchi en [89] propone un método de segmentación jerárquico en el cual los enlaces son removidos iterativamente teniendo en cuenta su "coeficiente de clustering", el cual se define como la razón entre el número de recorridos que usan este enlace y el mayor número posible de recorridos que pueden usar el enlace. A partir de ahora, vamos a denotar este procedimiento como algoritmo-*Radicchi*.

Newman y Girvan en [80] introducen el concepto de "modularidad" que se utiliza para determinar si una partición es adecuada (calidad de la partición). Dada una partición $\mathcal{P}$ de una red, la modularidad se define como

$$Q_{\mathcal{P}} = \frac{1}{2m} \sum_{i,j} \left[A(i,j) - \frac{k_i \; k_j}{2m} \right] \delta(c_i \; c_j), \tag{1.1}$$

donde m es la suma de todos los pesos asociados a los enlaces en el grafo, k_i es el grado de valor del nodo i, A es la matriz de adyacencia y $\delta(c_i \; c_j)$ es igual a 1 si los nodos i y j pertenecen al mismo grupo, y 0 en caso contrario. La modularidad de una partición representa la fracción de aristas que caen dentro de los grupos dados menos su fracción esperada si las aristas fueran distribuidas al azar.

Para obtener una representación matricial de la modularidad, definimos a S_{ir} tomando el valor de 1 cuando el nodo i pertenece al grupo r, y 0 en caso contrario. Luego, el término $\delta(c_i \; c_j)$ en 1.1 como:

$$\delta(c_i \; c_j) = \sum_r S_{ir} S_{jr}.$$

Así, la modularidad puede escribirse como:

$$\begin{aligned} Q_{\mathcal{P}} &= \frac{1}{2m} \sum_{i,j} \sum_r \left[A(i,j) - \frac{k_i \; k_j}{2m} \right] S_{ir} S_{jr} \\ &= \frac{1}{2m} \mathrm{Tr} \Big(\mathbf{S^T B S} \Big) \end{aligned} \tag{1.2}$$

donde la matriz $\mathbf{S}$ tiene por componentes $\mathbf{S} = \big\{ S_{ir} \big\}$, y la matriz $\mathbf{B}$ es llamada "matriz de modularidad", cuyos elementos son $\mathbf{B} = \Big\{ A(i,j) - \frac{k_i \; k_j}{2m} \Big\}$.

Algoritmos aglomerativos

La idea original fue propuesta en 2004 por Newman (ver [78]). Algunas mejoras fueron implementadas por Clauset, Newman y Moore en [23], en lo que llamaremos el algoritmo-CNM. La idea de estos algoritmos es producir un agrupamiento jerárquico de la red de una manera de aglomeración (es decir, a partir de los nodos aislados y acabado con un solo grupo). Esencialmente, los pasos son los siguientes:

- Calcular el aumento de la modularidad para cada posible unión en la red
- Seleccionar la combinación que maximiza el incremento de la modularidad y fusionar ambas comunidades.
- Repetir hasta que quede una sola comunidad.

Algoritmos aleatorios

Estos algoritmos se basan en la simulación de un proceso de difusión en el grafo, por ejemplo, el algoritmo de clustering de Markov, desarrollado en [100]. Este algoritmo es ampliamente usado en bioinformática; sin embargo, como es señalado en [68], es muy difícil de implementar ya que el rendimiento del algoritmo depende en gran medida algunos de los parámetros que se tienen que determinar. En el marco de algoritmos aleatorios, otro procedimiento importante en la detección de comunidades es el comúnmente llamado algoritmo-*Walktrap*. Este algoritmo se basa en caminos aleatorios para medir la similitud entre nodos (ver [88] para más detalles). Una vez que la similitud entre nodos adyacentes ha sido medida, los nodos pueden ser unidos. a partir de ahora, este algoritmo será denotado como algoritmo **Walktrap**, el cual describe a continuación:

Con el fin de agrupar los nodos en comunidades, Walktrap en [88] introduce una distancia r entre nodos que trata de capturar la estructura de las comunidades en el grafo. Esta distancia debe ser mayor si los dos nodos pertenecen a comunidades distintas, en caso contrario; cuando los nodos pertenezcan a la misma comunidad, la distancia debe ser pequeña. Este valor es calculado a partir de la información entregada por "paseos aleatorios" en el grafo.

Consideremos paseos aleatorios en el grafo G de una longitud dada t. Usaremos la información proporcionada por las probabilidades P_{ij}^t relacionadas con ir de i a j en t pasos. La longitud t del paseo aleatorio debe ser lo suficientemente largo para reunir información suficiente acerca de la topología del grafo. Para comparar dos vértices i y j utilizando estos datos, debe tenerse en cuenta lo siguiente:

- Si dos nodos i y j están en la misma comunidad, la probabilidad P_{ij}^t seguramente será alta. Sin embargo, que la probabilidad P_{ij}^t sea alta, no necesariamente implica que i y j estén en la misma comunidad.
- La probabilidad P_{ij}^t está influenciada por el grado k_j debido al hecho que paseo aleatorio tiene una alta probabilidad de ir a los nodos de grado alto.
- Dos nodos de una misma comunidad tienden a "ver" al resto de los nodos de la misma manera. Es decir, si los nodos i y j pertenecen a la misma comunidad, probablemente se tenga que $P_{ik}^t = P_{jk}^t$.

Teniendo en cuenta lo anterior, se da a continuación una definición de distancia entre nodos:

Definición 1.1: *Sean i y j dos nodos en un grafo G, entonces definimos una distancia, r_{ij}, para dichos nodos como*

$$r_{ij} = \left(\sum_{l=1}^{n} \frac{(P_{il}^t - P_{jl}^t)^2}{k_l} \right)^{1/2}.$$

Definición 1.2: *Sean C_1 y C_2 dos comunidades. Se define una distancia entre estas dos comunidades como:*

$$r_{C_1,C_2} = \left(\sum_{l=1}^{n} \frac{(P_{C_1 l}^t - P_{C_2 l}^t)^2}{k_l} \right)^{1/2}.$$

Una vez hechas estas definiciones, podemos describir el proceso del algoritmo de Walktrap de la siguiente forma:

1. Se empieza desde la partición $\mathcal{P}_1 = \{\{v\}, v \in V\}$ del grafo en que las n comunidades han sido reducidas a un sólo nodo. En primer lugar, calculamos las distancias entre todos los nodos adyacentes. A continuación, esta partición evoluciona mediante la repetición de las operaciones (2-4):

2. Se eligen dos comunidades C_1 y C_2 en $\mathcal{P}_k$ de acuerdo a un criterio basado en la distancia entre comunidades,

3. fusionar estas dos comunidades en una nueva comunidad $C_3 = C_1 \cup C_2$, y crear la nueva partición $\mathcal{P}_{k+1} = (\mathcal{P}_k \ \{C_1, C_2\}) \cup \{C_3\}$, y

4. actualizar las distancias entre comunidades (adyacentes).

Después de $n - 1$ iteraciones, el algoritmo termina y se obtiene $\mathcal{P}_n = \{V\}$. Cada paso define una partición $\mathcal{P}_k$ del grafo en comunidades, las cuales dan una estructura jerárquica (dendograma).

Observación

En el Capítulo 2 propondremos una técnica no supervisada de agrupación a través de grafos, la cual tratará de dar solución al problema de detección de comunidades a través de un enfoque jerárquico, utilizando como base la construcción de árboles soporte asociados al grafo de la red en estudio, de tal forma que permitan realizar una "buena" detección de comunidades en la red.

1.4. Segmentación de imágenes

La segmentación de imágenes separa una imagen en varias regiones que cumplen ciertas propiedades en términos de unas reglas predefinidas. Las regiones pueden representar objetos, parte de un objeto, o fondo. Uno de los objetivos fundamentales en la segmentación de imágenes es encontrar regiones de interés en una imagen de acuerdo a alguna aplicación particular.

Vista como una de las más importantes técnicas en visión artificial (o computacional), la segmentación de imágenes es ampliamente usada una gran variedad de aplicaciones, tales como reconocimiento biométrico (rostro humano, manos o huellas), procesamiento de imágenes médicas, análisis de imágenes por satélite, sistemas de control de tráfico (conteo de vehículos o reconocimiento de la matrícula del vehículo), edición fotográfica digital, visión robótica, entre otras.

Las diferentes técnicas o enfoques de segmentación de imágenes incluyen métodos basados en clustering [73], compresión [90], detección de bordes [95], histogramas [82], ecuaciones diferenciales parciales [120], transformación de cuencas [58], grafos [7], conjuntos borrosos [11, 12, 13, 14, 15, 16], entre otros.

1.4.1. Segmentación de imágenes en los últimos 50 años

La segmentación de imágenes es una importante técnica de tratamiento de imágenes conocida por su utilidad y complejidad. Para extraer información útil de las imágenes o grupos de imágenes, un paso inevitable es separar los objetos del fondo. La segmentación es el proceso o técnica adecuada para realizar esta tarea y muchas más. La segmentación de imágenes es a menudo descrita como el proceso que divide una imagen en sus partes constituyentes (Objetos). Es una de las tareas más críticas en el análisis automatizado de la imagen, que es la "capa central" de la *Ingeniería de imagen*. La ingeniería de imagen, está compuesta por tres capas a saber: procesamiento de la imagen (capa 1), análisis de la imagen (capa 2), y comprensión (o entendimiento) de la imagen (capa 3). La ingeniería de imagen es una nueva disciplina y proporciona un marco general para las técnicas de imágenes (ver [65, 122]).

La historia de la segmentación de imágenes digitales utilizando ordenadores se remonta a los años 60's. En 1965, se introdujo el "Operador de Roberts", un operador para la detección de los bordes entre las diferentes partes de una imagen; también conocido como 'Detector de Borde de Roberts' (ver [91]). Desde entonces, el campo de la segmentación de imágenes se ha desarrollado muy rápidamente y ha experimentado grandes cambios (ver [121]).

Una de las primeras definiciones formales de segmentación de imágenes (ver [41]), supone que toda la imagen R puede ser representada por un conjunto de regiones R_i, $i = 1, 2, \ldots, n$ no vacías y disjuntas en R, que cumplen las siguientes condiciones:

I. $\bigcup_{i=1}^{n} R_i = R$.

II. Para todo i y j tal que $i \neq j$, se cumple que $R_i \cap R_j = \emptyset$.

III. Para todo $i \in \{1, \ldots, n\}$, se debe cumplir que $P(R_i) =$ "CIERTO"; donde $P(R_i)$ es alguna *afirmación de uniformidad* para todos los elementos del conjunto R_i.

IV. Para todo $i \neq j$, $P(R_i \cup R_j) =$ "FALSO".

V. Para todo $i \in \{1, \ldots, n\}$, R_i está conectado (en grafos: R_i es una componente conexa).

Cada una de las condiciones anteriores tiene un significado particular. La condición (I) señala que la unión de las regiones segmentadas deben incluir todos los píxeles de la imagen. La condición (II) hace referencia a que no existan solapamientos entre las diferentes regiones .La condición (III) indica que los píxeles de una misma región deben tener propiedades similares y la condición (IV) advierte que píxeles provenientes de diferentes regiones deben tener algunas propiedades que los diferencian. La condición (V) dice que los píxeles que pertenezcan a una misma región como resultado de la segmentación, están conectados.

1.5. Segmentación de imágenes basada en grafos.

Las técnicas de segmentación de imágenes que utilizan grafos se ha ido convirtiendo en un área de investigación bastante próspera en temas como visión por computador, entre otros. A continuación, hablaremos rápidamente de algunos de estos métodos.

Tabla 1.1: Categorización de algunas técnicas de segmentación basadas en grafos.

	Mínimo Corte – Máximo Flujo	**Paseo Aleatorio**	**Mínimo árbol soporte**	**Corte Normal**	**Partición Isoperimétrica**
Complejidad	$O(mn^2)$, con n número de nodos y m número de aristas	$O(n)$, donde n es el número de nodos.	$O(m \log m)$ con m número de aristas.	$O(mn)$, donde n es el número de pixeles y m el número de pasos para la convergencia.	$O(m + n \log n)$, con n número de nodos y m es el número de aristas.
Desempeño	Menos de un segundo para imágenes 2D de 512×512	Alrededor de 3 segundos en imágenes de 256 para un Intel Xeon 2.4GHz con 3GB RAM	No hay información disponible	2 minutos en un Intel Pentium 200MHz para imágenes de 100×200	0.58 segundos en una AMD Athlon de 1.4GHz con 512k RAM
Imágenes de prueba	Imágenes médicas y secuencias	Imágenes médicas 2D y 3D, base de datos de Berkeley	Escenas de la Naturaleza, base de datos de Columbia COIL	Escenas naturales e imágenes de radar del clima	Escenas de la naturaleza
Aplicaciones	Edición de fotos y vídeo, segmentación de imágenes médicas	Segmentación de imágenes médicas y objetos	Segmentación de objetos e imágenes	Segmentación en escenas de la naturaleza	Segmentación de escenas de la naturaleza, imágenes médicas

En términos matemáticos, se hablará siempre de un grafo $G = (V, E)$ compuesto por un conjunto de nodos $v \in V$ y un conjunto de aristas $e \in E$ que conectan dos nodos $v_i \in V$ y $v_j \in V$ (es decir, $e = (v_i, v_j) \in V \times V$). En los procesos de segmentación de imágenes a través de grafos, un nodo puede ser un pixel o un conjunto de pixeles con alguna característica en común; y los valores asignados a las aristas (que relacionan nodos) describen similaridades o disimilaridades entre los nodos que conecta de acuerdo a alguna regla específica. Una categorización de algunas técnicas de segmentación basadas en grafos se muestra en la Tabla 1.1. A continuación se describirá brevemente algunos de estos métodos agrupándolos según sean técnicas supervisadas o no supervisadas.

1.5.1. Métodos Supervisados

En los Métodos de segmentación supervisados se hace relevante la interacción de los usuarios, ya sea a través del marcado de algunos pixeles pertenecientes al "objeto" y al "fondo", o dibujando un rectángulo que abarque la totalidad del objeto que se desea segmentar, entre otros tipos de interacción. En esta clase de métodos se puede obtener lo que se conoce como el "primer plano" y el 'fondo" que se desee. De los cinco modelos analizados en la Tabla 1.1, el modelo de "Mínimo Corte/Máximo Flujo" y el modelo del "Paseo Aleatorio" están en esta clase de procesos supervisados.

Mínimo Corte - Máximo Flujo

En el modelo de segmentación de "mínimo corte-máximo flujo" (también llamado "corte de grafo"), una imagen está representada por un grafo no dirigido $G = (V, E)$. Dos términos adicionales especiales se incluyen en el grafo, usualmente llamados la "fuente" (objeto), denotado por s, y el "fondo" denotado por t. Sean P y N un con-

junto de píxeles de una imagen y el conjunto de todos los pares no ordenados $\{v_1, v_2\}$ de los píxeles vecinos en P, respectivamente. Si $A = (A_1, \ldots, A_p, \ldots, A_{|P|})$ denota un vector binario en el cual cada componente es una etiqueta A_p asignada a un píxel p representando ya sea "objetos" ("obj") o "fondo" ("bkg"). Así el vector binario A es el resultado de la segmentación. Entre todas las posibles segmentaciones, la segmentación óptima se encuentra incorporando algunas restricciones fuertes y débiles. Las restricciones débiles son los bordes y las propiedades regionales de segmentación A se describen por la función de coste definida como:

$$E(A) = \lambda \cdot R(A) + B(A) \tag{1.3}$$

donde

$$R(A) = \sum_{p \in P} R_p(A_p) \tag{1.4}$$

$$B(A) = \sum_{\{p,q\} \in N} B_{\{p,q\}} \cdot \delta\Big(A_p, A_q\Big) \tag{1.5}$$

y

$$\delta\Big(A_p, A_q\Big) = \begin{cases} 1 & \text{si } A_p \neq A_q \\ 0 & \text{en otro caso} \end{cases} \tag{1.6}$$

En la Ecuación (1.3), $R(A)$ y $B(A)$ dados en (1.4) y (1.5), suelen llamarse término regional (o término "dato") y término de frontera (o término de "suavidad") respectivamente. λ es un escalar no negativo para ajustar la compensación entre los términos de región y de frontera. En la Ecuación (1.4), $R_p(A_p)$ denota la penalización para la asignación de A_p al pixel p. En la ecuación (1.5), $B_{\{p,q\}}$ denota la penalización por discontinuidad entre p y q. Las restricciones fuertes son impuestas por la interacción del usuario. En el método de segmentación interactiva, un tipo popular de restricciones fuertes es que algunos píxeles se marcan como "obj" y otros cuantos píxeles se marcan como "bkg" por un usuario:

$$\begin{aligned} &\forall p \in \mathcal{O}, && A_p = \text{"obj"} \\ &\forall p \in \mathcal{B}, && A_p = \text{"bkg"} \end{aligned} \tag{1.7}$$

donde $\mathcal{O}$ y $\mathcal{B}$ representan el conjunto de los píxeles marcados como "obj" y "bkg", respectivamente, y tal que $\mathcal{O} \subset P$ y $\mathcal{B} \subset P$ con $\mathcal{O} \cap \mathcal{B} = \emptyset$. Siguiendo las restricciones fuertes, los píxeles restantes sin marcar se asignan a "obj" o "bkg" en la segmentación óptima $A_{\text{óptima}}$.

$A_{\text{óptimo}}$ minimiza globalmente la función de coste dada en la Ecuación (1.3) entre todas las posibles segmentaciones "objeto/fondo" para una imagen dada. El mínimo corte $C_{\text{óptimo}}$ es un subconjunto de E que bi-particiona el grafo correspondiente a $A_{\text{óptimo}}$. Un soporte teórico adicional acerca de la existencia de la segmentación óptima definida por el corte mínimo minimizando la función de coste (1.3), entre todas las segmentaciones que cumplan las restricciones fuertes en la Ecuación (1.7), han sido probadas en [7].

Algunas investigaciones se han centrado en la modificación del algoritmo original de Mínimo corte - Máximo flujo, con el fin de obtener algoritmos más eficientes con mejores segmentaciones; véase por ejemplo [8]. Un estudio sobre los modelos de flujo

máximo en términos continuos en lugar de la situación discreta, es presentado en Yuan et al. en 2010 (ver [113]). La selección del parámetro λ en la función de coste (1.3) no es trivial, debido a que valores diferentes de λ afectan significativamente el resultado de la segmentación. Peng et al. en 2008 (ver [85]), observaron que valores diferentes del parámetro λ pueden producir sobre-segmentación, baja-segmentación, incluso buenas segmentaciones; además han propuesto un algoritmo supervisado de selección automática del parámetro que intenta obtener la mejor segmentación para cada imagen.

Algoritmo del paseo aleatorio

El algoritmo iterativo y multinivel del paseo aleatorio para segmentación de imágenes (random walker algorithm), fue descrito inicialmente por Grady en 2004 (ver [54, 55]). Según esta teoría, un camino es una secuencia de nodos y aristas, iniciando en un nodo y terminando también con un nodo, y sin restricciones sobre el número de veces que un nodo puede ser visitado. Los pesos de las aristas del grafo son tratados como una probabilidad en el algoritmo del paseo aleatorio. Por otro lado, también se calcula la probabilidad de que un pixel, no etiquetado, alcance a alguno de los pre-marcados (etiquetados) por el paseo aleatorio. Si un pixel no etiquetado tiene una probabilidad mayor de alcanzar a algún pre-etiquetado con L que a cualquier otro pre-etiquetado, entonces a este pixel se le asigna la etiqueta L. De esta forma, la segmentación se lleva a cabo mediante la asignación de todos los pixeles no marcados a una de las etiquetas.

Con el fin de mejorar el tiempo computacional de este algoritmo, Grady en 2008 (ver [57]) propuso un pre-proceso a la segmentación, antes del marcado de las "semillas" para el fondo y los objetos por parte del usuario. También se desarrolló una aproximación lineal para el algoritmo del paseo aleatorio a través de los vectores propios asociados a la matriz Laplaciana de los pesos del grafo. Con estas dos mejoras, el algoritmo modificado mostraba ser alrededor de 14 veces más rápido que el algoritmo original del paseo aleatorio (ver [56]). En el caso particular de imágenes médicas, también se han propuesto técnicas para acelerar este proceso (ver [3]).

1.5.2. Métodos no Supervisados

Métodos de segmentación no supervisados dividen una imagen en varias regiones basados en algunas métricas predefinidas, y sin interferencia del usuario. El método de mínimo árbol soporte, el método de corte normalizado y el método isoperimétrico, son algunos de los métodos de segmentación no supervisados.

Mínimo Árbol Soporte

Felzenszwalb en 2004 (ver [36]), definió la segmentación de una imagen como una partición de todos los nodos en componentes de forma tal que cada componente de la segmentación corresponde a una componente conexa. Además define la frontera entre dos componentes y también define la "diferencia interna"de una componente como el mayor peso en el mínimo árbol soporte de dicha componente. La diferencia entre dos componentes se define como el menor peso entre las aristas que conectan las dos componentes. Si la diferencia entre dos componentes es mayor que la mínima diferencia interna de las dos componentes, se considera que hay una frontera entre dichas componentes,

de lo contrario, no existe frontera entre las dos componentes. Como este algoritmo considera propiedades locales, también suele llamarse "segmentación de imágenes basado en variación local"

Método de Corte Normalizado

Shi en 2001 (ver [96]), transfirió el problema de segmentación de una imagen a un problema de partición de grafos. Para evitar el corte en pequeños grupos de nodos como lo hace el "Proceso de Corte Mínimo" propuesto en 1993 por Wu (ver [110]), el corte normalizado, N_{CUT}, fue propuesto como un criterio global para evaluar la disparidad entre diferentes grupos y la homogeneidad entre grupos.

El corte normalizado de dos conjuntos A y B en un grafo $G = (V, E)$, tales que $A \subset V$, $B \subset V$, $A \cup B = V$ y $A \cap B = \emptyset$, se define como:

$$N_{\mathrm{CUT}}(A, B) = \frac{\mathrm{cut}\big(A, B\big)}{\mathrm{assoc}\big(A, V\big)} + \frac{\mathrm{cut}\big(A, B\big)}{\mathrm{assoc}\big(B, V\big)} \tag{1.8}$$

donde $\mathrm{cut}(A, B)$ es el grado de similitud entre A y B, $\mathrm{assoc}(A, V)$ es la conexión total de A en el grafo y $\mathrm{assoc}(B, V)$ es la conexión total de B en el grafo; y se calculan así:

$$\mathrm{cut}\big(A, B\big) = \sum_{u \in A, v \in B} w\big(u, v\big) \tag{1.9}$$

$$\mathrm{assoc}\big(A, V\big) = \sum_{u \in A, t \in V} w\big(u, t\big) \tag{1.10}$$

$$\mathrm{assoc}\big(B, V\big) = \sum_{s \in B, t \in V} w\big(s, t\big). \tag{1.11}$$

donde $w(u, v)$ es el peso de la arista que conecta a u y v.

El problema de minimización del corte normalizado se transformó entonces al de solucionar un sistema de autovalores. El vector propio asociado con el segundo valor propio más pequeño, en relación al sistema de autovalores, es el que se usa para la bi-partición del grafo. La segmentación final se obtiene recursivamente en las partes segmentadas. En 2009, Xu (ver [111]) combinó el algoritmo de corte normalizado con el conocimiento a priori de ciertas restricciones para mejorar el rendimiento del algoritmo de regmentación (en realidad, utilizó un algoritmo iterativo garantizando su convergencia, pues no se podía incorporar las restricciones al método de corte normalizado original). El algoritmo propuesto anteriormente ha sido usado en otros métodos de corte en grafos (ver [21, 63]).

Partición Isoperimétrica

Siguiendo el problema problema isoperimétrico clásico en geometría (que afirma que dada un área fijada, encontrar la región con perímetro mínimo), Grady en 2006 (ver [56]) propuso un algoritmo isoperimétrico para segmentación de imágenes dividiendo la imagen en regiones con las áreas mas grandes y perímetros más pequeños. Dado un

grafo $G = (V, E)$, la razón isoperimétrica de un conjunto isoperimétrico $S \subset G$ se define como:

$$h(S) = \inf_{S} \frac{|\partial S|}{\mathrm{Vol}_S} \tag{1.12}$$

En la ecuación (1.12), la frontera de S se define como

$$\partial S = \left\{ e_{ij} \;\mid\; v_i \in S, \;\; v_j \in S^c \right\}$$

con S^c el complemento de S. Además

$$\left|\partial S\right| = \sum_{e_{i,j} \in \partial S} w(e_{ij}) \qquad \text{y} \qquad \mathrm{Vol}_S = \sum_i d_i \quad \forall v_i \in S,$$

donde $w(e_{ij}) = w_{ij}$ es el peso de e_{ij}, y d_i es el grado del vértice v_i, con $d_i = \sum_{e_{ij}} w_{ij}$ para todo $e_{ij} \in E$.

El objetivo del algoritmo isoperimétrico es maximizar Vol_S y minimizar $\left|\partial S\right|$. En contraste con el algoritmo de corte normalizado, el de partición isoperimétrica del grafo es más rápido y estable.

Observación

La técnica que propondremos a lo largo de este trabajo se puede clasificar dentro de los procesos no supervisados de segmentación a través de grafos, y tratará de dar solución al problema de dividir una red a través de un enfoque jerárquico, utilizando como base la construcción de árboles soporte asociados al grafo de la red en estudio, y exigiendo que dichos árboles soporte cumplan algunas condiciones particulares, de tal forma que permitan realizar una "buena" partición de la red.

Capítulo 2

Segmentación Jerárquica en Redes. Enfoque D&L

Contenido

Es este Capítulo definiremos formalmente el concepto el "Problema de Partición Jerárquica en Redes" (**HCNP**, siglas en inglés), para lo que se requiere introducir el concepto de partición, partición fina y partición jerárquica de un grafo, el cual nos permitirá diferenciar y clasificar los algoritmos de partición de redes. También se propone el algoritmo Divide-and-Link, (D&L), el cual tratará de dar solución al problema HCNP para una red cualquiera, y que está enmarcado en los procesos de partición de redes a través de grafo. Este Capítulo está organizado de la siguiente manera: en la Sección 2.1 formalizaremos el problema segmentación jerárquica de una red; luego, en la Sección 2.4 se muestra una manera novedosa para resolver el problema planteado en la Sección 2.1. Este algoritmo se basa en un procedimiento iterativo binario, obtenido de acuerdo a una familia de umbrales (también conocidos como niveles o cotas) previamente fijados. Una apropiada selección de dicha familia de umbrales define el algoritmo

para la segmentación jerárquica de la red Esta jerarquizadación puede visualizarse por medio de un dendrograma, el cual se construye fácilmente a partir de la ordenación de los nodos que define el procedimiento iterativo binario anterior.

2.1. Segmentación jerárquica en redes

Muchos de los problemas reales, y en particular los problemas de redes sociales, puede modelarse usando teoría de grafos: es decir, a través de nodos y aristas. Los nodos del grafo pueden corresponden a "individuos"; y si dos individuos están unidos por una arista es porque pueden relacionarse entre sí directamente (por ejemplo, si viven en el mismo edificio, pertenecen a la misma organización que trabajan o son miembros de un mismo foro de Internet). En general, fijemos la siguiente notación:

- $V = \{1, 2, \ldots, n\}$ representará un conjunto (finito) de elementos que se desea agrupar (clustering).
- $E = \big\{ \{i, j\} \quad | \quad i, j \in V \big\}$ será el conjunto de pares de nodos (o vecinos) de V; esto es, si dos elementos $i, j \in V$ están relacionados, entonces existe una arista $e = \{i, j\} \in E$; de lo contrario, $\{i, j\} \notin E$. Además, el número de aristas será representado por m (es decir, $m = |E|$).
- Dado $e = \{i, j\} \in E$, sea $d_e \geq 0$ el grado de disimilitud entre sus puntos finales i y j: mientras más grande sea d_e más disimilitud habrá entre i y j. La medida de disimilitud es definida teniendo en cuenta el problema específico y las características de los elementos considerados (la construcción y propiedades de tales medidas de afinidad son un tópico que no será tratado de manera directa en este capítulo.)

Por consiguiente, hemos definido un grafo $G = (V, E)$ que muestra las relaciones entre sus nodos. El grafo G puede asumirse conexo; pues de no serlo, entonces cada componente conexa puede ser analizada separadamente. Finalmente, la información disponible puede por lo tanto puede ser resumida por la red N:

$$N = \Big(G = (V, E) \; ; \; \{d_e \; , \; e \in E\} \Big). \tag{2.1}$$

Dada una red, uno de los principales objetivos es detectar grupos o estructuras cohesivas, y por ende, uno de los principales problemas es cómo hacerlo. Dar solución a este problema equivale a realizar una partición de la red en grupos (o comunidades, para el caso de redes sociales).

En la actualidad, se puede encontrar en la literatura una variedad de enfoques cuyo objetivo primordial es identificar los grupos (o comunidades) inherentes en una red (para más detalles, ver por ejemplo [37]). Estos algoritmos de clustering sobre redes detectan las comunidades de acuerdo a las estructuras topológicas o comportamiento dinámico de las mismas.

Los algoritmos de segmentación a través de grafos pueden clasificarse dependiendo de la forma en que cada uno de ellos realiza la agrupación. Fortunato [37], basado en las diferentes técnicas de agrupación, establece una división entre los métodos tradicionales, agrupación espectral, los algoritmos divisivos, métodos basados en modularidad y algoritmos dinámicos.

En este capítulo nos centraremos en algoritmos de segmentación jerárquica. La principal ventaja de este enfoque es que se reproduce una evolución (que puede representarse con un dendograma) de cómo se forman los grupos (métodos aglomerativos) o cómo se fraccionan (métodos divisivos), desde el inicio del proceso hasta el paso final. Aunque los métodos de segmentación jerárquica pueden traer más problemas computacionales, tienen una ventaja importante (sobre todo en el análisis de redes sociales), ya que los resultados son más informativos que los dados por los algoritmos no jerárquicos, que sólo proporcionan una "imagen final" del proceso.

Las transiciones entre las "familias" podrán ser entonces visualizadas por medio de la dendograma obtenido a partir de la agrupación jerárquica. De esta manera no sólo puede elegir una segmentación apropiada para cualquier valor seleccionado (o umbral), sino que también podemos obtener una mejor comprensión de la propia red.

La familia de clases obtenidas de esta forma puede ser vista como el primer paso hacia una segmentación o clasificación posterior, en función del objetivo final de nuestro análisis.

Los problemas de segmentación en redes (por ejemplo, en detección de comunidades), son usualmente definidos como el problema de hallar una *buena* partición de un grafo dado $G = (V, E)$.

Definición 2.1 (Partición de un grafo (ver [37])): *Dado un grafo $G = (V, E)$, una "partición" $\mathcal{P}$ del grafo G se define como un conjunto $\mathcal{P} = \{C_1, \ldots, C_r\}$ que cumple:*

1. $\bigcup_{j=1}^{r} C_j = V$, *(cubrimiento),*
2. $C_i \cap C_j = \emptyset$ *para todo* $i \neq j$ *(disjuntos), y*
3. *para todo i, el subgrafo $(C_i, E_{|C_i})$ es conexo.*

Ahora bien, para introducir el concepto de partición jerárquica, definamos primero el concepto de *partición fina*. Note que la condición (2) de la definición anterior, para el caso de redes borrosas, no necesariamente se satisface.

Definición 2.2 (Partición fina): *Sean $\mathcal{P}$ y $\mathcal{Q}$ dos particiones de un grafo $G = (V, E)$. Se dice que $\mathcal{P}$ es más fina que $\mathcal{Q}$ (y lo denotamos $\mathcal{P} \widetilde{\subseteq} \mathcal{Q}$) si para todo $A \in \mathcal{P}$, existe un $B \in \mathcal{Q}$, tal que $A \subseteq B$.*

Definición 2.3 (Partición jerárquica): *Consideremos el grafo $G = (V, E)$, y sea $\boldsymbol{\mathcal{D}} = (\mathcal{P}^0, \mathcal{P}^1, \mathcal{P}^2, \ldots, \mathcal{P}^r)$ una secuencia de particiones para dicho grafo. Diremos que $\boldsymbol{\mathcal{D}}$ es una partición jerárquica de G (obtenida de forma divisiva) si se cumple que:*

1. *Existen dos particiones triviales: la primera ($\mathcal{P}^0$), que agrupa a todos los nodos; y la última ($\mathcal{P}^r$), compuesta de 'singuletes', es decir, compuesta de grupos con un nodo. En símbolos, $\mathcal{P}^r = \big\{ \{i\}, \ i \in V \big\}$.*
2. *$|\mathcal{P}^i| > |\mathcal{P}^{i-1}|$ para todo $i = 1, \ldots, r$ (es decir, en cada iteración se incrementa el número de grupos o comunidades).*
3. *$\mathcal{P}^i \widetilde{\subseteq} \mathcal{P}^{i-1}$ para todo $i = 1, \ldots, r$ ($\mathcal{P}^i$ más fina que $\mathcal{P}^{i-1}$).*

El ejemplo que viene a continuación ilustra el concepto de partición jerárquica, además muestra cuándo es válida la Definición 2.3 y cuando no lo es.

Ejemplo 2.1.
Consideremos el grafo $G = (V, E)$, donde el conjunto de nodos es $V = \{1, 2, 3, 4\}$ y con conjunto de aristas $E = \big\{\{1, 2\}, \{1, 3\}, \{2, 4\}, \{3, 4\}\big\}$. Una forma de representación para el grafo G es dada en la Figura 2.1.

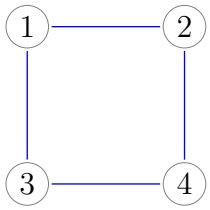
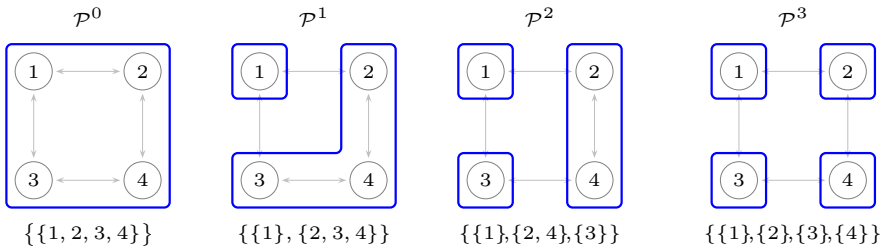


Figura 2.1: Grafo $G = (V, E)$

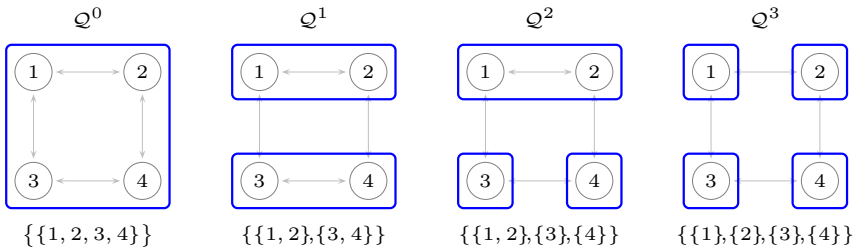
Algunas posibles particiones para el grafo mostrado en la Figura 2.1 son:

Partición 1: $\mathcal{P}^0 = \Big\{\{1, 2, 3, 4\}\Big\}$, $\mathcal{P}^1 = \Big\{\{1\}, \{2, 3, 4\}\Big\}$, $\mathcal{P}^2 = \Big\{\{1\}, \{2, 4\}, \{3\}\Big\}$ y $\mathcal{P}^3 = \Big\{\{1\}, \{2\}, \{3\}, \{4\}\Big\}$.



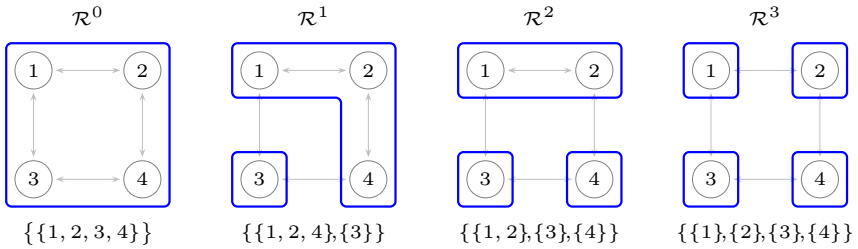
Note que $\boldsymbol{\mathcal{D}}_1 = \{\mathcal{P}^0, \mathcal{P}^1, \mathcal{P}^2, \mathcal{P}^3\}$ es una partición jerárquica para G.

Partición 2: $\mathcal{Q}^0 = \Big\{\{1, 2, 3, 4\}\Big\}$, $\mathcal{Q}^1 = \Big\{\{1, 2\}, \{3, 4\}\Big\}$, $\mathcal{Q}^2 = \Big\{\{1, 2\}, \{3\}, \{4\}\Big\}$, y $\mathcal{Q}^3 = \Big\{\{1\}, \{2\}, \{3\}, \{4\}\Big\}$.



Note que $\boldsymbol{\mathcal{D}}_2 = \{\mathcal{Q}^0, \mathcal{Q}^1, \mathcal{Q}^2, \mathcal{Q}^3\}$ es una partición jerárquica para G.

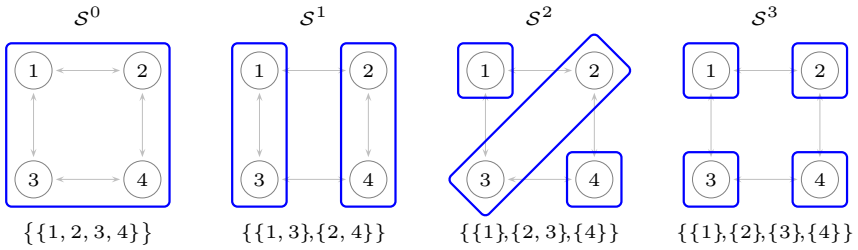
Partición 3: $\mathcal{R}^0 = \Big\{\{1, 2, 4, 3\}\Big\}$, $\mathcal{R}^1 = \Big\{\{1, 2, 4\}, \{3\}\Big\}$, $\mathcal{R}^2 = \Big\{\{1, 2\}, \{3\}, \{4\}\Big\}$ y $\mathcal{R}^3 = \Big\{\{1\}, \{2\}, \{3\}, \{4\}\Big\}$.



Note que $\boldsymbol{\mathcal{D}}_3 = \{\mathcal{R}^0, \mathcal{R}^1, \mathcal{R}^2, \mathcal{R}^3\}$ es una partición jerarquizada de G.

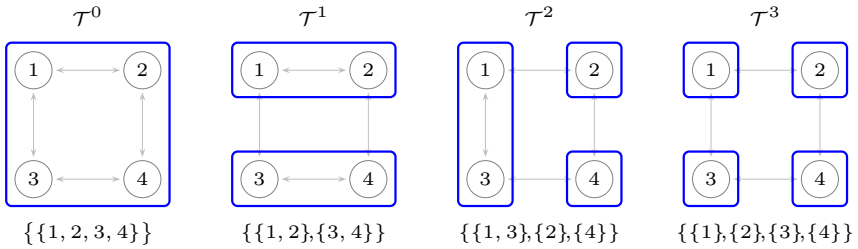
A continuación, mostraremos dos secuencias asociadas al grafo G de la Figura 2.1 que sin embargo, no cumplen la definición de partición jerárquica:

Partición 4: $\mathcal{S}^0 = \Big\{\{1,2,3,4\}\Big\}$, $\mathcal{S}^1 = \Big\{\{1,3\},\{2,4\}\Big\}$, $\mathcal{S}^2 = \Big\{\{1\},\{2,3\},\{4\}\Big\}$ y $\mathcal{S}^3 = \Big\{\{1\},\{2\},\{3\},\{4\}\Big\}$.



En este caso, la secuencia $\boldsymbol{\mathcal{D}}_4 = \{\mathcal{S}^0, \mathcal{S}^1, \mathcal{S}^2, \mathcal{S}^3\}$ no es jerárquica debido a que los nodos $\{2\}$ y $\{3\}$ no están conectados en el grafo original G y en consecuencia, no se puede hablar de partición. (pues en $\mathcal{S}^2$, el subgrafo $\{2,3\}$ no es conexo).

Partición 5: $\mathcal{T}^0 = \Big\{\{1,2,3,4\}\Big\}$, $\mathcal{T}^1 = \Big\{\{1,2\},\{3,4\}\Big\}$, $\mathcal{T}^2 = \Big\{\{1,3\},\{2\},\{4\}\Big\}$ y $\mathcal{T}^3 = \Big\{\{1\},\{2\},\{3\},\{4\}\Big\}$.



En la secuencia $\boldsymbol{\mathcal{D}}_5 = \{\mathcal{T}^0, \mathcal{T}^1, \mathcal{T}^2, \mathcal{T}^3\}$ te tiene que $\mathcal{T}^2 \,\widetilde{\not\subseteq}\, \mathcal{T}^1$, por lo cual no es una partición jerárquica.

A continuación, se estudia un concepto fundamental en partición de grafos, conocido como "coloración de grafos". En problemas relacionados con particiones de grafos, tiene sentido pensar en coloraciones del grafo, pues una coloración induce una partición del grafo.

2.2. Coloración de grafos

Definición 2.4 (*c*-coloración): *Una c-coloración de un grafo $G = (V, E)$ es una aplicación*

$$f : V \longrightarrow \{0, 1, \ldots, c-1\}. \tag{2.2}$$

Así, si $v \in V$, $f(v)$ es llamado el "color" de v.

Observaciones:

- Cualquier c-coloración induce una partición del conjunto V, siendo cada clase asociada a un color:
$$V_C(k) = \{i \in V \mid f(i) = k\} \tag{2.3}$$
con $k \in \{0, 1, \ldots, c-1\}$.
- Toda coloración induce una partición de V, pero no siempre se obtiene una partición del grafo como dada por la Definición 2.1.
- Un caso particular de la c-coloración (2.2) es el coloreado binario, que identificaremos en adelante por col, y viene dado por:
$$\text{col} : V \longrightarrow \{0, 1\} \tag{2.4}$$
- Un proceso sucesivo de coloreado binario sobre el grafo G, induce un partición jerárquica, en el sentido que cualquier subconjunto de V puede ser particionado en dos: aquellos coloreados con cero, y aquellos coloreados con 1. Este proceso se detalla a continuación.

Dado un nodo $i \in V$ coloreado binariamente, $(\text{col}(i) \in \{0, 1\})$, el color de cualquier nodo adyacente $j \in V$ dependerá de la medida d_e, $(e = \{i, j\} \in E)$, cuando es comparada con un umbral fijado α:

$$\text{col}(j) = \begin{cases} \text{col}(i) & \text{si } d_e < \alpha \\ 1 - \text{col}(i) & \text{si } d_e \geq \alpha \end{cases} \tag{2.5}$$

para todo $e \in E$. Note que (2.5) es una regla de coloreado que separa dos nodos (coloreándolos con diferentes colores) según el "**grado de disimilaridad**" entre ellos.

Ejemplo 2.2.
Considérese el grafo mostrado en la Figura 2.2. A continuación, vamos a colorearlo según la regla (2.5) para un valor particular de α, digamos $\alpha = 2$.

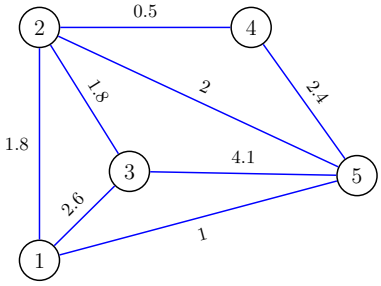


Figura 2.2: Grafo valuado

Para colorearlo, usemos un árbol soporte asociado al grafo. La idea es aprovechar la estructura misma del árbol soporte para asignarles el valor de '0' o '1' a cada nodo, y conseguir un coloreado binario. Hay que observar que se elige la estructura de árbol para colorear de forma única, puesto que si existiera un ciclo, el color de un nodo podría variar dependiendo de la cadena de coloración utilizada, como se verá más adelante.

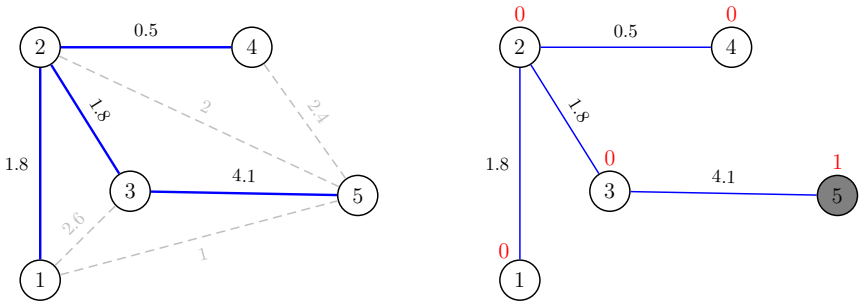


Figura 2.3: Árbol soporte y coloración de los nodos, para $\alpha = 2$.

En la Figura 2.3 se muestra un posible árbol soporte asociado a grafo inicial, y a continuación se muestra el coloreado de cada nodo (comenzando, sin pérdida de generalidad, con la asignación de '0' para el nodo marcado con el número 1). Recordemos que el valor del umbral es $\alpha = 2$. Sin embargo, ésta no es la única posibilidad de colorear el grafo. Unas cuantas posibilidades más son dadas en la Figura 2.4 (nótese que el resultado no es siempre el mismo, pues depende del árbol soporte elegido).

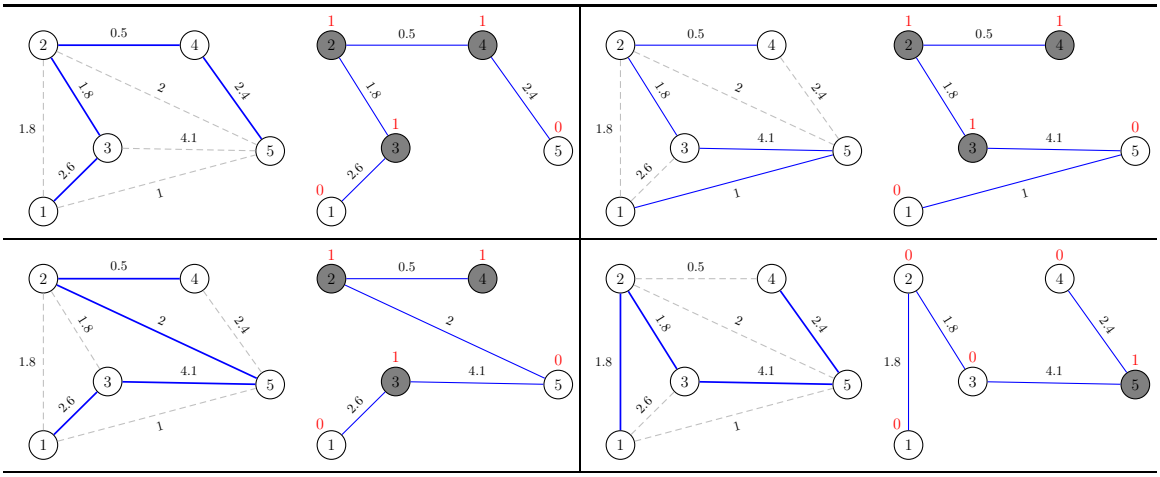


Figura 2.4: Coloreados usando diferentes árboles soporte, para $\alpha = 2$.

Este tipo de problemas relacionados con el coloreado de un grafo, que resulta con frecuencia el grafo tiene ciclos, será tratado con más profundidad más adelante. Por ahora vamos a establecer la estructura del modelo jerarquizado que deseamos implementar.

2.3. Algoritmo Jerárquico Simple

A continuación, proponemos un algoritmo de segmentación jerárquica que mejora el algoritmo propuesto en [49], y que hemos en la descrito en la Sección 2.2. Consideremos

una red N definida en (2.1) y una familia de valores de umbrales

$$\alpha_1 > \alpha_2 > \ldots > \alpha_K \tag{2.6}$$

donde $\alpha_1 = \max_{e \in E}\{d_e\}$ y $\alpha_K = \min_{e \in E}\{d_e\}$. Definimos la siguiente familia de grafos parciales:

$$\left\{ G^i = (V, E^i) \right\} \tag{2.7}$$

con $i \in \{1, 2, \ldots, K\}$, y donde $E^i = \{e \in E \mid d_e \geq \alpha_i\}$. Construyamos la familia de bosques soporte $F^i = (V, T^i)$ (un bosque es un grafo sin ciclos, sus componentes conexas son árboles) con los siguientes dos pasos:

1. Sea $F^i = (V, T^i)$ el mínimo bosque soporte de los grafos parciales $G^i | G^{i+1} = (V, E^i - E^{i+1})$. Siguiendo un algoritmo similar al de Kruskal, la construcción de F^i es basada en una ordenación de las aristas cuyos pesos pertenecen al intervalo $(\alpha_i, \alpha_{i+1}]$ en orden decreciente.

2. Si F^i así construido es ya un bosque soporte de $G^i | G^{i+1}$, terminar el proceso; en otro caso, entonces las aristas por fuera de E^i (aristas menores que α_i) serán ordenadas en orden creciente, y luego añadidas iterativamente a T^i (si no forman ciclos), y finalizamos el proceso cuando F^i sea el bosque soporte.

Observaciones.

- Los bosques F^i no son ni los máximos ni los mínimos bosques soportes, pues el orden en el cual las aristas son adicionadas es invertido en el segundo paso.
- La formación de subgrupos homogéneos para un valor α determinado, tiene en cuenta la estructura de los subgrupos formados por el α anterior.
- Esta propiedad es muy útil en el tratamiento de diversos problemas, en particular, cuando se desean segmentaciones jerarquizadas.

El coloreado binario separa dos nodos (asignándoles un color diferente) si el grado de similaridad entre ellos es menor que el valor fijado α. Cuando este valor decrece, es más estricta la partición del conjunto de nodos. Luego, una familia de valores $\left\{\alpha_1, \ldots, \alpha_K\right\}$ puede entonces producir una agrupación jerarquizada de V en el sentido en el cual dos nodos separados no podrán ser unidos en adelante. La búsqueda de estos parámetros α_i no es una tarea trivial. Este problema es estudiado en detalle en [50] y [49].

Este coloreado binario puede comenzar asignando un valor ("0" o "1") a un nodo arbitrario, y fijando el orden en el cual los otros nodos adyacentes van a ser coloreados por medio de la regla dada en (2.5). Este orden es muy importante y será analizado a continuación.

Nótese que el coloreado binario anterior no siempre es consistente: un nodo puede ser coloreado de forma diferente, dependiendo de la cadena escogida (secuencias de aristas adyacentes) unida con el primer nodo. Estas situaciones de inconsistencia se presentan cuando existen ciclos en el grafo. Estos ciclos son llamados "ciclos de inconsistencia".

Obviamente, si el grafo es acíclico, no habrían ciclos de inconsistencia y ésto justifica el uso de un árbol soporte para controlar el proceso de coloreado binario. Por otro lado, si

el grafo tiene ciclos (que es el caso más frecuente), tales inconsistencias pueden aparecer al de aplicar el algoritmo de coloreado binario.

Dado un árbol soporte genérico, los ciclos de inconsistencia pueden aparecer si se añaden aristas. El número de tales inconsistencias dependen del árbol soporte y del umbral α escogidos. En determinados problemas (ver, por ejemplo, [49]), cuando se ocupan de los ciclos de inconsistencia en imágenes digitales reales, y teniendo en cuenta la gran cantidad de nodos en la red, una razón pequeña de tales ciclos de inconsistencia es permitida sin perder calidad en el proceso de segmentación. Más aún, la topología particular de la red de nodos permite algunos atajos en el arreglo de nodos (ver [49], para más detalles). Otro desarrollo para operar con estos ciclos de inconsistencia es introducido en [112], donde el menor árbol soporte es definido para cualquier valor de α; pero este desarrollo tiene un alto costo computacional y no asegura la consistencia del proceso de coloreado binario.

La partición jerárquica será definida a través de una sucesión de árboles soporte variando los valores del umbral α desde el valor más grande hasta el más pequeño que pueda tomar en el grafo. En cada iteración, el proceso de coloreado binario será controlado por estos árboles soporte, y el grafo original será cambiado borrando aquellas aristas con pesos mayores. Inicialmente, todos los nodos pertenecen al mismo cluster; en la primera iteración los nodos son agrupados en dos clases: los coloreados con 0 y los coloreados con 1; cada grupo será particionado de manera similar con la propiedad que dos nodos separados en una iteración no podrán ser unidos posteriormente. Dicho proceso finaliza cuando todos los nodos están aislados.

Si el número de coloreados binarios sucesivos es muy grande, la partición pudiera ser excesivamente fina y no informativa. Una forma de resolver esto es reducir el parámetro K. Este parámetro se puede escoger teniendo en cuenta que, por un lado, la partición debe considerar valores diferentes de las distancias y, por otro lado, el costo computacional para obtener la partición debe reducirse. Además, luego de fijar el valor de K, se deben escoger los diferentes valores de $\alpha_1 > \alpha_2 > \cdots > \alpha_K$. Estas dos decisiones se toman dependiendo del tamaño y características del problema.

Ejemplo 2.3.
Considere el grafo general dado en la Figura 2.5. Aplicaremos el proceso de segmentación jerarquizada a dicho grafo usando un conjunto de valores para α:

$$\{\alpha = 6{,}2, \quad \alpha = 3, \quad \alpha = 1{,}2, \quad \alpha = 1, \quad \alpha = 0\}$$

Para obtener la segmentación jerarquizada para este conjunto de umbrales, empezaremos en orden decreciente, es decir; $\alpha \in \{6{,}2 \geq 3 \geq 1{,}2 \geq 1 \geq 0\}$. El proceso se muestra a continuación:

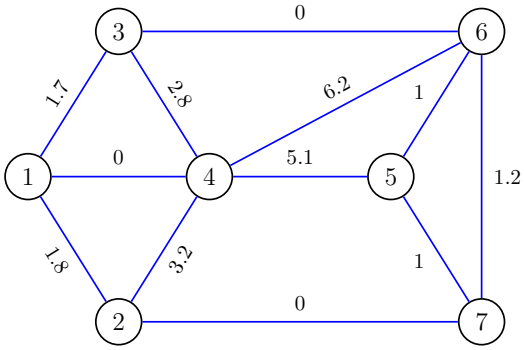
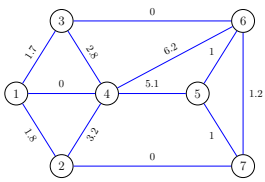
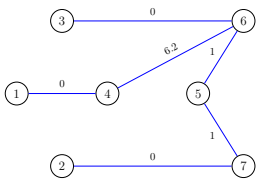
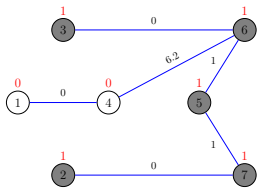
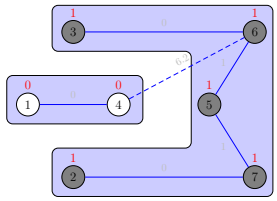
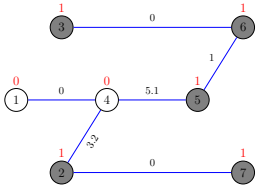
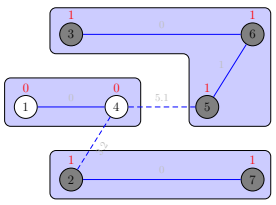
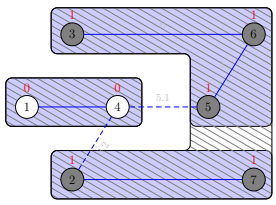
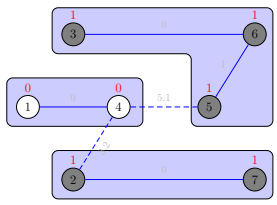


Figura 2.5: Grafo valuado

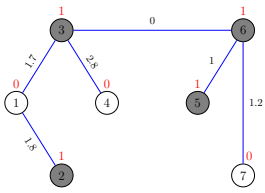
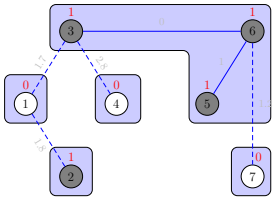
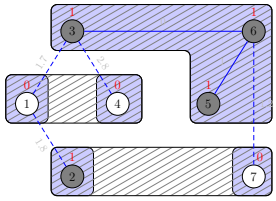
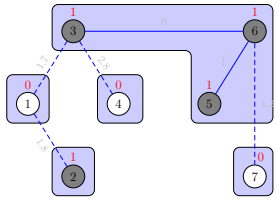
Para $\alpha = 6{,}2$:

			
Grafo original.	Obtención del árbol soporte a partir de la inclusión de aristas, donde $\alpha = 6{,}2$.	Coloreado de nodos obtenido a partir del umbral $\alpha = 6{,}2$.	Formación de grupos homogéneos, a partir del coloreado.

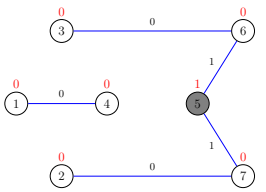
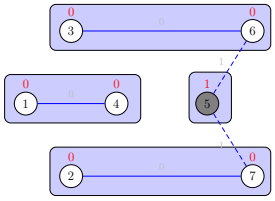
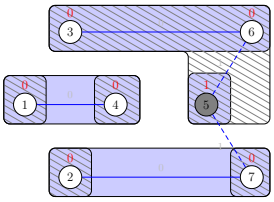
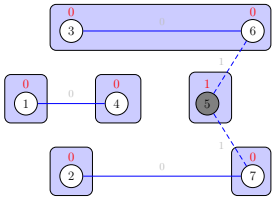
Para $\alpha = 3$:

			
Obtención del árbol soporte para $\alpha = 3$.	Formación de grupos homogéneos a partir del coloreado.	Comparación de las regiones para $\alpha = 3$ respecto a $(\alpha = 6{,}2)$.	Grupos finales para $\alpha = 3$ (resultado de la intersección).

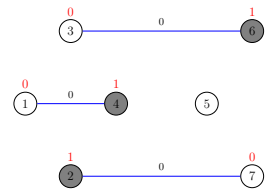
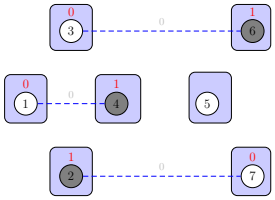
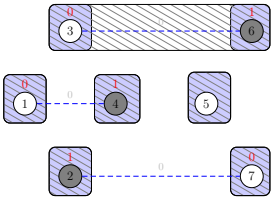
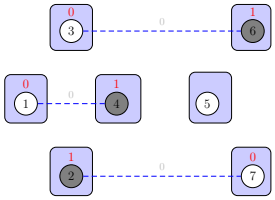
Para $\alpha = 1{,}2$:

			
Obtención del árbol soporte para $\alpha = 1{,}2$.	Formación de grupos homogéneos a partir del coloreado.	Comparación de las regiones para $\alpha = 1{,}2$ respecto a $(\alpha = 3)$	Grupos finales para $\alpha = 1{,}2$ (resultado de la intersección)

Para $\alpha = 1$:

			
Construcción del árbol soporte para $\alpha = 1$.	Formación de grupos homogéneos a partir del coloreado obtenido.	Comparación de las regiones obtenidas a partir de $\alpha = 1$ respecto al anterior ($\alpha =$ 1,2)	Grupos finales para $\alpha = 1$ (resultado de la intersección)

Para $\alpha = 0$:

			
Bosque soporte obtenido a partir de la inclusión de aristas, teniendo como referencia $\alpha = 0$.	Formación de grupos homogéneos a partir del coloreado obtenido.	Comparación de las regiones obtenidas a partir de $\alpha = 0$ respecto al anterior ($\alpha =$ 1)	Grupos definitivos para $\alpha = 0$ (resultado de la intersección)

En resumen, se tienen la siguiente secuencia de segmentación jerarquizada (incluyendo un valor de $\alpha = 7$ en el cual todos los nodos trivialmente pertenecen al mismo grupo):

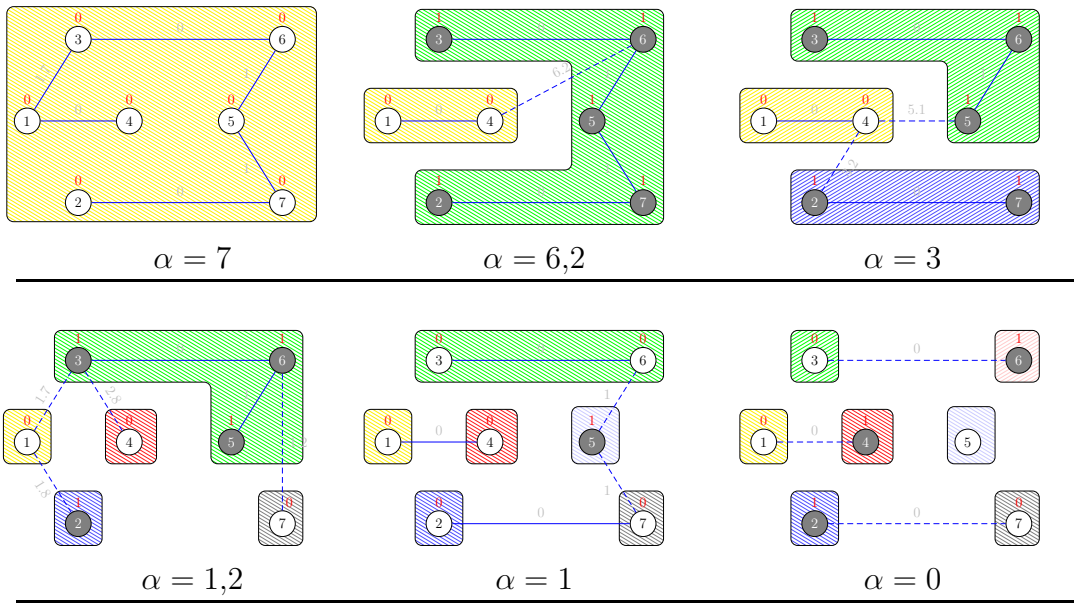


Figura 2.6: Segmentación jerárquica para el Ejemplo 2.3.

El algoritmo de segmentación jerárquica que hemos descrito en esta sección puede verse como una introducción a un procedimiento más general para realizar un clustering jerárquico de una grafo, y que hemos llamado "Proceso de Segmentación Jerárquica Divide-and-Link", o de manera más simple, "Algoritmo Divide-and-Link" o también "Algoritmo D&L". Este procedimiento se describe a continuación:

2.4. Enfoque "Divide-and-Link" (D&L)

El proceso de agrupación (clustering) que definiremos aquí comparte varios elementos claves del algoritmo de segmentación propuesto en [49], y el cual es extendido en [48] dentro del marco de clasificación en imágenes. Ese algoritmo fue desarrollado inicialmente para clasificar píxeles de una imagen, con la finalidad de identificar regiones homogéneas en una clasificación no supervisada. Su principal objetivo era obtener una segmentación de la imagen digital en estudio. Sin embargo, el algoritmo en [48] era válido solamente para una clase específica de la red. Esta limitación será superada y, además, la complejidad computacional del algoritmo de segmentación propuesto en [48] también será mejorada.

Nuestro objetivo es agrupar el conjunto V del grafo $G = (V, E)$ por medio de un procedimiento iterativo binario que clasifica los nodos de V en dos clases: V_0 y V_1. Esta clasificación depende del tipo de aristas: los puntos extremos de las **aristas de división** serán asignados a diferentes clases, y los puntos finales de las **aristas de enlace** serán asignados a la misma clase. De esta manera, el primer tipo de aristas *dividen* los nodos y el segundo tipo de aristas *enlazan* los nodos.

Este procedimiento iterativo induce una partición jerárquica del conjunto G. Por ejemplo, la clase $V_0 \subset V$ se dividirá de nuevo en dos clases V_{00} y V_{01}, y así sucesivamente.

En cualquier etapa de este proceso jerárquico, $E_d \subset E$ representa el conjunto de *aristas de división*, y $E_l \subset E$ representa el conjunto de *aristas de enlace*. Dado un nodo clasificado $i \in V_0$, si la arista $e = \{i, j\} \in E_d$, entonces el nodo j será clasificado en una clase diferente $j \in V_1$; sin embargo, si la arista $e = \{i, j\} \in E_l$, entonces el nodo j será clasificado misma clase $j \in V_0$.

El núcleo del algoritmo *Divide-and-Link* será el proceso que clasifica las aristas en E_d o E_l. En este proceso se tendrán en cuenta dos etapas:

- **La etapa de división**: Las aristas serán ordenadas en orden decreciente de divisibilidad, es decir, se comienza con la arista que más divide hasta terminar con la que menos divide; no importa si se trata de un valor numérico o categórico, sólo es necesario que las aristas puedan ordenarse (el orden no es estricto, así que los empates entre aristas están permitidos).

  Esta primera etapa depende del problema a resolver, por ejemplo, esta etapa podría estar asociada con el concepto de *intermediación* (betweenness), ampliamente utilizado en el contexto de las redes sociales; o también, podría asociarse con la diferencia de *intensidad* en el contexto de las imágenes astronómicas. En general, denotaremos con d_e (para $e \in E$ en el grafo $G = (V, E)$) a la medida de *división* que será usada, dependiendo del problema.

- **La etapa de enlace**: Las aristas son ordenadas en orden decreciente de similitud, es decir, desde la más afín hasta la menos afín. Igual que en la etapa de división, los empates también son permitidos. La medida asociada a esta etapa se denota como l_e, y se construye para todas las aristas $e \in E$. Valores grandes de l_e, implica que sus puntos finales de $e \in E$ deben estar más cercanos en el proceso de segmentación.

De todos modos, las medidas d_e y l_e son, respectivamente, las medidas de disimilitud y la afinidad definidas en el conjunto V, las cuales pueden ser, dependiendo del problema, independientes de la estructura del grafo $G = (V, E)$.

El algoritmo de segmentación jerárquica que proponemos es divisivo, comienza con la partición trivial $\mathcal{P}^0 = \{V\}$ y termina con la otra partición trivial que consta de tantos grupos como nodos hay en el conjunto V. Este proceso reduce el conjunto inicial de aristas $E^0 = E$ en cada iteración t, introduciendo grafos parciales $G^t = (V, E^t)$ con la propiedad de que $E^{t+1} \subset E^t$ para todo t, y terminando cuando $E^K = \emptyset$. Los pasos principales del algoritmo Divide-and-Link son descritos en la Tabla 2.1.

Tabla 2.1: Descripción del enfoque Divide-and-Link.

1.	Se calculan los valores (pesos) de división y de enlace para cada arista.
2.	Las aristas del grafo parcial G^t se disponen secuencialmente según los pesos calculados anteriormente.
3.	Basados en la disposición anterior, y siguiendo un algoritmo similar al de Kruskal, se construye un bosque soporte, $F^t \subset G^t$.
4.	La partición $\mathcal{P}^t$ es construida siguiendo el procedimiento binario, el cual es aplicado sobre F^t.
5.	El nuevo conjunto de aristas E^{t+1} se define a partir de E^t, eliminando las aristas que unen los diferentes grupos de $\mathcal{P}^t$.
6.	Repetir los pasos (1)-(5) siempre que $E^{t+1} \neq \emptyset$.

En la Sección 2.6 se encuentra el pseudocódigo general del algoritmo de segmentación jerárquica Divide-and-Link. Los pasos principales de este proceso jerárquico serán detallados a continuación.

2.4.1. Estructura jerárquica: número de niveles y valores de umbrales

El proceso jerárquico se llevará a cabo mediante una elección adecuada del parámetro K, el número de niveles jerárquicos, y los valores de los parámetros $\alpha_1, \alpha_2, \ldots, \alpha_K$ que impulsarán el proceso de agrupación en cada nivel. Los valores de α se mueven en el dominio de la medida divisiva d_e y están acotados por:

$$\left\{ \overline{d} = \alpha_1 > \alpha_2 > \cdots > \alpha_K = \underline{d} \right\}$$

donde $\underline{d} \equiv \min_{e \in E} \left\{ d_e \right\}$ y $\overline{d} \equiv \max_{e \in E} \left\{ d_e \right\}$.

2.4.2. El grafo parcial $G^t = (V, E^t)$ y la disposición de aristas

Inicialmente, cuando $t = 0$, $E^0 = E$, así $G^0 = (V, E^0) = G$ es el grafo inicial. La partición inicial es la trivial $\mathcal{P}^0 = \left\{C_1^0\right\}$ con $C_1^0 = V$.

En general, en el paso $t \in \{1, 2, \ldots, K\}$ es conocido el conjunto E^{t-1} de aristas y la partición V: $\mathcal{P}^{t-1} = \left\{ C_1^{t-1}, C_2^{t-1}, \ldots, C_{s_{t-1}}^{t-1} \right\}$ la cual verifica $\bigcup_{i=1}^{s_{t-1}} C_i^{t-1} = V$ y $C_i^{t-1} \cap C_j^{t-1} = \emptyset$ para todo $i \neq j$. El conjunto E^t será construido como un subconjunto de E^{t-1} eliminando aquellas aristas que unan grupos diferentes:

$$E^t = E^{t-1} - E_f \tag{2.8}$$

donde, el conjunto E_f en (2.8) viene dado por:

$$E_f \equiv \left\{ e = \{a, b\} \in E^{t-1} \;\middle|\; a \in C_i^{t-1},\;\; b \in C_j^{t-1};\;\; i \neq j; \quad i, j \in \{1, 2, \ldots, s_{t-1}\} \right\}$$

Para el paso inicial $t = 1$, no existen grupos diferentes en $\mathcal{P}^0$, lo que implica que $E_f = \emptyset$ y $E^1 = E^0$.

2.4.3. El bosque soporte $F^t = (V, W^t)$

Las aristas de división pertenecientes al conjunto $E_d = \left\{e_1, e_2, \ldots, e_{n_1}\right\} \subset E^t$, están indexadas de forma tal que verifican la siguiente desigualdad:

$$d_{e_1} \geq d_{e_2} \geq \cdots \geq d_{e_{n_1}} \geq \alpha_t.$$

Por otro lado, el conjunto de aristas de enlace: $E_l = \left\{e_{n_1+1}, e_{n_1+2}, \ldots, e_{n_1+n_2}\right\} \subset E^t$ están indexadas de forma tal que verifican las siguientes dos propiedades:

1. $d_{e_i} < \alpha_t \qquad$ para todo $\quad i \in \{n_1 + 1, n_1 + 2, \ldots, n_1 + n_2\}$

2. $l_{e_{n_1+1}} \geq l_{e_{n_1+2}} \geq \cdots \geq l_{e_{n_1+n_2}}$.

Finalmente, se construye un único listado de aristas, L, con la siguiente estructura ordenada:

$$L = E_d \cup E_l = \left\{ e_1, e_2, \ldots, e_{n_1}, e_{n_1+1}, \ldots, e_{n_1+n_2} \right\}. \tag{2.9}$$

Aplicando un algoritmo similar al de Kruskal al listado de aristas $L \subset E^t$ dado en (2.9), se construye un bosque soporte $F^t = (V, W^t) \subset T^t = (V, E^t)$: comenzando por $W^t = \emptyset$, las aristas de la lista L si irán incluyendo una a una en W^t, siempre y cuando la arista a incluir no forme un ciclo. Los componentes conexas del grafo parcial $F^t = (V, W^t)$ son árboles; sin embargo, no son los árboles soporte máximos ni mínimos debido a que el criterio de ordenación de las aristas se ha cambiado, pues una parte va según el criterio de división y otra parte según el criterio de enlace.

2.4.4. La partición $\mathcal{P}^t$

Dado el bosque soporte F^t, se tienen dos enfoques (equivalentes) para la construcción de la partición $\mathcal{P}^t$:

- **Conceptualmente:**

  Dado que $F^t = (V, W^t)$ es el bosque soporte de G^t; consideremos el grafo parcial $H^t = (V, W_l^t) \subset F^t$, donde $W_l^t = W^t \cap E_l$, es decir, las aristas de W^t las cuales son aristas de enlace.

  La nueva partición será $\mathcal{P}^t = \{C_1^t, C_2^t, \ldots, C_{s_t}^t\}$, donde los conjuntos C_i^t son las componentes conexas del grafo parcial H^t.

- **Algorítmicamente, a través de un proceso de coloreado binario:**

  Para cualquier componente conexa de F^t, sea $v \in V$ cualquier nodo arbitrario, el cual puede ser clasificado (coloreado) como $(\mathrm{col}^t(v) = 0)$, donde $\mathrm{col}^t : V \longrightarrow \{0, 1\}$ es la función de coloreado binario definida sobre V. Entonces, podemos seguir un esquema de propagación en cada componente conexa, desde los nodos coloreados a los nodos adyacentes, hasta que todos los nodos sean coloreados:

$$\mathrm{col}^t(v') = \begin{cases} \mathrm{col}^t(v) & \text{si } e = \{v, v'\} \in E_l \\ 1 - \mathrm{col}^t(v) & \text{si } e = \{v, v'\} \in E_d \end{cases} \tag{2.10}$$

  para todo $e = \{v, v'\} \in W^t$, donde $v, v' \in V$.

  Este proceso de coloreado binario induce una partición $\mathcal{P}^t = \{C_1^t, C_2^t, \ldots, C_{s_t}^t\}$ del conjunto V, donde los conjuntos $C_i^t \subset V$ son las componentes conexas de los subgrafos F^t generados por

$$V_0^t = \Big\{ v \in V \mid \mathrm{col}^t(v) = 0 \Big\} \qquad \text{y} \qquad V_1^t = \Big\{ v \in V \mid \mathrm{col}^t(v) = 1 \Big\}.$$

2.4.5. Jerarquización de las particiones

Siguiendo los pasos previos, las agrupaciones definidas por las sucesivas particiones $\mathcal{P}^t$ son jerárquicas, en el sentido que $\mathcal{P}^t$ es más fina que $\mathcal{P}^{t-1}$ (ver Definición 2.2). En efecto, para cualesquiera $t \in \{1, 2, \ldots, K\}$ e $i \in \{1, 2, \ldots, s_t\}$ existe un único $j \in \{1, 2, \ldots, s_{t-1}\}$ tal que $C_i^t \subset C_j^{t-1}$.

Una forma de demostrar esta propiedad es de manera constructiva. Sea $v \in C_i^t$ un nodo arbitrario de este conjunto que pertenece a la partición $\mathcal{P}^t$; como $\mathcal{P}^{t-1}$ es por construcción, una partición de V; entonces existe un único $j \in \{1, 2, \ldots, s_{t-1}\}$ tal que $v \in C_j^{t-1}$. Por otra parte, no hay ningún otro nodo $v' \in C_i^t$ con $v' \neq v$ tal que $v' \notin C_j^{t-1}$: en este caso, la arista $\{v, v'\} \in E_f$ en la iteración $t - 1$, y no podría pertenecer al conjunto E^t. En consecuencia, cualquier nodo de C_i^t pertenece a C_j^{t-1} y $C_i^t \subset C_j^{t-1}$.

Note que las particiones obtenidos son independientes del color en particular (0 ó 1) elegido para los nodos iniciales de cualquier componente conexa, puesto que en cada iteración dos nodos que provengan del mismo grupo compartirán el mismo color en la siguiente iteración (ya sea 0 ó 1).

2.4.6. El dendograma

Luego de realizar todo el proceso anterior, podemos construir un dendograma fácilmente a partir del orden natural inducido por el algoritmo Divide-and-Link, de tal manera que se pueda obtener una agrupación jerárquica de la red..

Sea $\{v_{(1)}, v_{(2)}, \ldots, v_{(n)}\}$ una lista lexicográficamente ordenada de los nodos de V:

$$\left(\mathrm{col}^1(v_{(i)}), \mathrm{col}^2(v_{(i)}), \ldots, \mathrm{col}^t(v_{(i)})\right) \prec^l \left(\mathrm{col}^1(v_{(j)}), \mathrm{col}^2(v_{(j)}), \ldots, \mathrm{col}^t(v_{(j)})\right) \qquad \forall i \leq j.$$

Con esta disposición se tiene entonces que dos nodos pertenecientes al mismo grupo en el nivel α_t ($v, v' \in C_k^t \in \mathcal{P}^t$, implica que $\mathrm{col}^s(v) = \mathrm{col}^s(v')$ para todo $s \in \{1, \ldots, t\}$) serán divididos en dos grupos al nivel α_{t+1} si $\mathrm{col}^{t+1}(v) \neq \mathrm{col}^{t+1}(v')$.

2.4.7. Complejidad Computacional

La complejidad computacional de nuestro algoritmo jerárquico Divide-and-Link (D&L) es $O = (df)$ donde d es el número de niveles de jerarquización, acotado por $n = \left|V\right|$ (note que, el caso extremo $d = n$ se obtiene cuando $\left|\mathcal{P}^i\right| = \left|\mathcal{P}^{i-1}\right| + 1$ para todo $t \in \{1, 2, \ldots, n\}$); y f es la máxima complejidad computacional de los siguientes procesos en cada nivel jerárquico (ver la Tabla 2.1):

1. Actualizar las aristas de división,
2. Actualizar las aristas de unión,
3. Ordenar las aristas
4. Calcular el bosque soporte,
5. Construir la partición,
6. Actualizar el conjunto de aristas.

Con el fin de ilustrar los conceptos y procedimientos introducidos en esta sección, presentaremos a continuación un ejemplo numérico simple.

Ejemplo 2.4.
Sea $V = \{1, 2, 3, 4, 5, 6, 7\}$ un conjunto de 7 elementos (nodos), y consideremos la red $N = \left\{ G = (V, E), \{d_e, l_e \;\; e \in E\} \right\}$ como se muestra en la Figura 2.7, donde, para cada nodo $e \in E$, el valor disimilitud d_e corresponde a la medida de intermediación (betweenness), y el valor de l_e es una medida de afinidad (en la Figura 2.7 se muestran las medidas de intermediación y afinidad, $d_{\{i,j\}}$ y $l_{\{i,j\}}$, calculadas usando las ecuaciones (3.7) y (3.8), respectivamente).

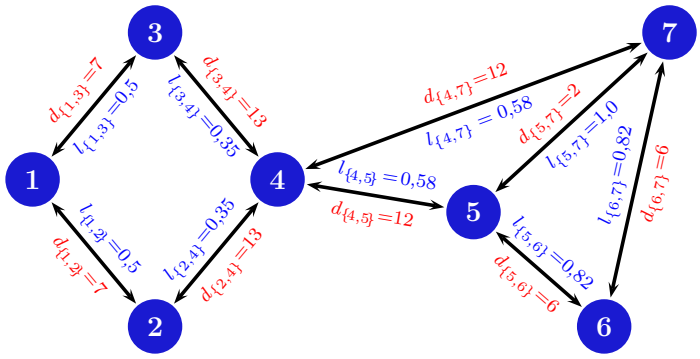


Figura 2.7: Red de 7 nodos, 9 aristas y dos medidas (disimilitud y afinidad) asociadas a cada arista.

Para la red mostrada en la Figura 2.7 se tienen 5 valores para la medida d_e. La familia completa de umbrales α es:

$$\overline{d} = \underset{(\alpha_1)}{13} > \underset{(\alpha_2)}{12} > \underset{(\alpha_3)}{7} > \underset{(\alpha_4)}{6} > \underset{(\alpha_5)}{2} = \underline{d} \tag{2.11}$$

El algoritmo Divide-and-Link usado con los umbrales (2.11),es desarrollado en los siguientes cinco pasos (uno por cada umbral):

Paso 1: $(i = 1) \quad \alpha_1 = 13.$

En este caso, $G^i = G^1 = G$, y F^1 es construido a partir del ordenamiento de las aristas cuya disimilitud sea menor o igual que $\alpha_1 = 13$ mayor que $\alpha_2 = 12$, $d_e \in (\alpha_2, \alpha_1]$, en orden decreciente:

$$\underbrace{\overset{\checkmark}{\underset{(d_{\{3,4\}})}{13}} \geq \overset{\checkmark}{\underset{(d_{\{2,4\}})}{13}}}_{E_d} \geq \underset{(d_{\{4,5\}})}{12} \geq \underset{(d_{\{4,7\}})}{12} \geq \underset{(d_{\{1,2\}})}{7} \geq \underset{(d_{\{1,3\}})}{7} \geq \underset{(d_{\{6,7\}})}{6} \geq \underset{(d_{\{5,6\}})}{6} \geq \underset{(d_{\{5,7\}})}{2} \tag{2.12}$$

Por lo tanto, las aristas de "división" son $E_d = \big\{\{3,4\},\{2,4\}\big\}$. En caso de empate, las aristas con menor l_e serán seleccionadas primero. Ambas aristas son incluidas en el bosque soporte F^1.

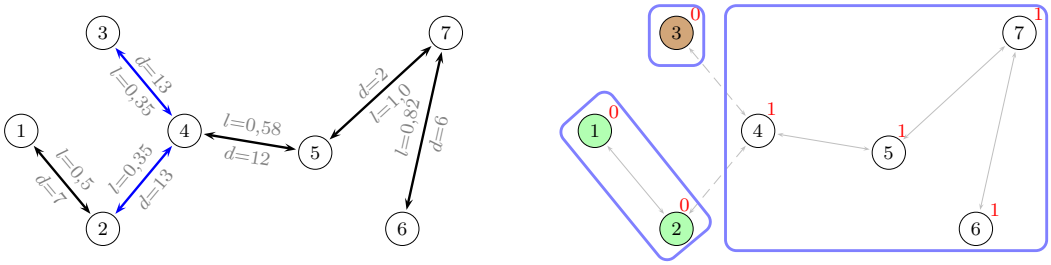
Las aristas restantes en (2.12) son ordenadas en orden decreciente, pero teniendo en cuenta los valores de afinidad l_e:

$$\underbrace{\overset{\checkmark}{\underset{(l_{\{5,7\}})}{1{,}0}} \geq \overset{\checkmark}{\underset{(l_{\{6,7\}})}{0{,}82}} \geq \underset{(l_{\{5,6\}})}{0{,}82} \geq \overset{\checkmark}{\underset{(l_{\{4,5\}})}{0{,}58}} \geq \underset{(l_{\{4,7\}})}{0{,}58} \geq \overset{\checkmark}{\underset{(l_{\{1,2\}})}{0{,}5}}}_{E_l} \geq \underset{(l_{\{1,3\}})}{0{,}5} \tag{2.13}$$

Siguiendo a (2.13), el bosque soporte F^1 es completado con las aristas (señaladas con ✓) $E_l = \big\{\{5,7\},\{6,7\},\{4,5\},\{1,2\}\big\}$. Note que las aristas $\{5,6\}$ y $\{4,7\}$ no son utilizadas para formar el bosque soporte, pues formarían ciclos. Además, $\{1,3\}$ no se utiliza debido a que el árbol ha sido terminado. El árbol soporte es mostrado en la Figura 2.8a. Las aristas marcadas con azul, son las aristas de división pertenecientes a E_d, las otras aristas son las de enlace que pertenecen a E_l. El primer nodo, nodo 1, es clasificado con

0; los otros nodos son clasificados dependiendo de si la arista con las que se llega a cada uno de ellos (en el árbol) pertenece a E_d o E_l.

Teniendo en cuenta si los nodos igualmente clasificados son conexos en G, se tienen tres grupos: $\{1,2\}$, $\{3\}$ and $\{4,5,6,7\}$ (ver la Figura 2.8b).



(a) (b)

Figura 2.8: (a) Árbol soporte $F^1 = (V, T^1)$, donde $T^1 = E_d \cup E_l$. (b) Grupos: $\{1,2\}$, $\{3\}$ y $\{4,5,6,7\}$.

Paso 2: $(i = 2)$ $\alpha_2 = 12$**.**

En este caso, $G^i = G^2$ y las aristas $\{3,4\}$ y $\{2,4\}$ son eliminadas de G^1. La arista $\{1,3\}$ es también eliminada de G^1 porque conecta dos grupos. F^2 es construido ordenando las aristas menores o iguales que $\alpha_2 = 12$ y mayores que $\alpha_3 = 7$, $d_e \in (\alpha_3, \alpha_2]$, en orden decreciente:

$$\underbrace{\overset{\checkmark}{12}_{(d_{\{4,5\}})} \geq \overset{\checkmark}{12}_{(d_{\{4,7\}})}}_{E_d} \geq 7_{(d_{\{1,2\}})} \geq 6_{(d_{\{6,7\}})} \geq 6_{(d_{\{5,6\}})} \geq 2_{(d_{\{5,7\}})} \tag{2.14}$$

Por lo tanto, las aristas de división son $E_d = \big\{\{4,5\},\{4,7\}\big\}$. En caso de empate, la arista con menor l_e será seleccionada primero. Ambas aristas son entonces incluidas en el bosque soporte F^2. Las aristas restantes en (2.14) son ordenadas en orden decreciente, pero teniendo en cuenta el criterio l_e:

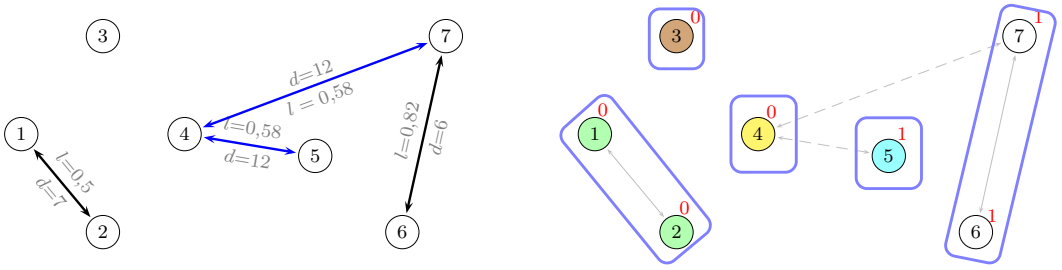
$$\underbrace{1{,}0_{(l_{\{5,7\}})} \geq \overset{\checkmark}{0{,}82}_{(l_{\{6,7\}})} \geq 0{,}82_{(l_{\{5,6\}})} \geq \overset{\checkmark}{0{,}5}_{(l_{\{1,2\}})}}_{E_l} \tag{2.15}$$

Siguiendo (2.15), el bosque soporte F^2 es completado con las aristas $E_l = \big\{\{6,7\},\{1,2\}\big\}$ (aristas marcadas con ✓).

El bosque soporte resultante es mostrado en la Figura 2.9a. Las aristas de color azul son las pertenecientes a E_d (las de división), las otras aristas son las pertenecientes a E_l. Teniendo en cuenta la agrupación obtenida en el Paso 1, el grupo $\{4,5,6,7\}$ es dividido en tres grupos: $\{4\}$, $\{5\}$ y $\{6,7\}$ (ver Figura 2.9b).

Paso 3: $(i = 3)$ $\alpha_3 = 7$**.**

$G^i = G^3$, las aristas $\{4,5\}$ y $\{4,7\}$ son eliminadas de G^2. Las aristas $\{5,6\}$ y $\{5,7\}$ son también eliminadas de G^2 porque conectan grupos diferentes $F^3 = (V, T^3)$



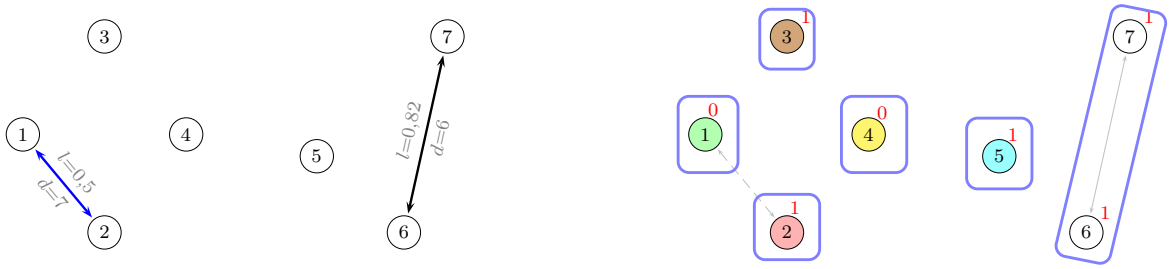
(a) (b)

Figura 2.9: (a) Bosque soporte $F^2 = (V, T^2)$. (b) Grupos: $\{1, 2\}$, $\{3\}$ $\{4\}$, $\{5\}$ y $\{6, 7\}$.

se construye como antes, donde:

$$\underbrace{\overset{\checkmark}{7}}_{E_d}\ \underset{(d_{\{1,2\}})}{\geq}\ \underset{(d_{\{6,7\}})}{6} \qquad \text{and} \qquad \underbrace{\overset{\checkmark}{1,2}}_{E_l}\ \underset{(l_{\{6,7\}})}{}$$

El bosque soporte F^3 y los grupos formados se muestran en la Figura 2.10.



(a) (b)

Figura 2.10: (a) Bosque soporte $F^3 = (V, T^3)$. (b) Grupos: $\{1\}$, $\{2\}$ $\{3\}$, $\{4\}$, $\{5\}$ y $\{6, 7\}$.

Paso 4: $(i = 4)$ $\alpha_4 = 6$**.**

$G^i = G^4$, la arista $\{1, 2\}$ es eliminada de G^3, y $F^4 = (V, T^4)$ se construye como antes, donde

$$\underbrace{\overset{\checkmark}{6}}_{E_d}\ \underset{(d_{\{6,7\}})}{} \qquad \text{and} \qquad E_l = \phi$$

El bosque soporte F^4 y los grupos formados se muestran en la Figura 2.11.

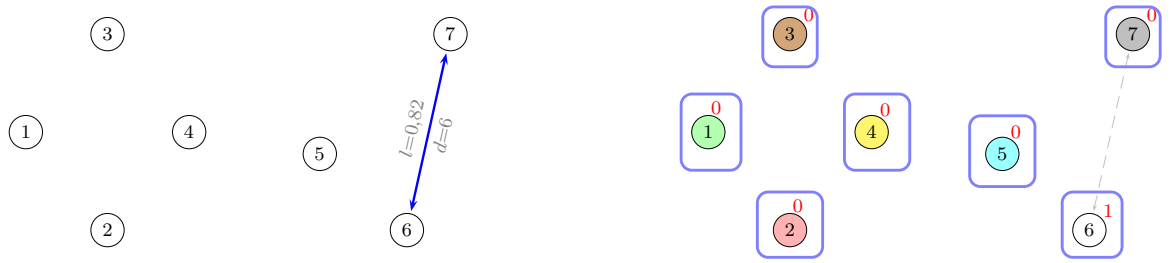
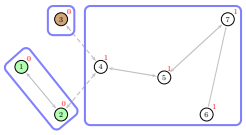
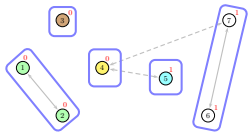
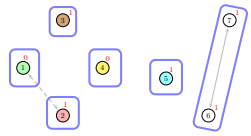
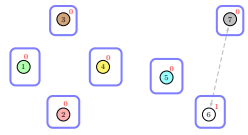


Figura 2.11: (a) Bosque soporte $F^4 = (V, T^4)$. (b) Grupos: {1}, {2}, {3}, {4}, {5}, {6} y {7}.

Paso 5: $(i = 5)$ $\alpha_5 = 2$. Este paso no es necesario, debido a que en el paso anterior (Paso 4) se produjo la máxima partición para este ejemplo.

Tabla 2.2: Resumen: división jerárquica para el Ejemplo 2.4

Paso 1.	Paso 2.	Paso 3.	Paso 4.
			

A continuación daremos una breve explicación de la información obtenida por la aplicación del algoritmo Divide-and-Link para la red N del Ejemplo 2.4.

El procedimiento binario iterativo que hemos introducido sugiere un proceso de partición jerárquica: los primeros grupos se definen por los componentes conexas de nodos igualmente clasificadas, y cualquiera de ellos se puede dividir de nuevo por el procedimiento binario posterior. De esta manera, los nodos separados no se pueden unir de nuevo en este proceso iterativo, por lo que las primeras etapas de este proceso jerárquico son más relevantes que las posteriores.

Note que las particiones que se producen son independientes de la clasificación particular (0 o 1) elegida, ya que en cada iteración dos nodos pertenecientes al mismo grupo, compartirán el mismo grupo en la siguiente iteración si comparten la misma clase, independientemente de si esta clase se asocia al 0 o 1.

Los cinco procesos jerárquicos binarios de la red N en el Ejemplo 2.4 y sus grupos asociados son mostrados en la Tabla 4.2.

Tabla 2.3: Partición jerárquica para el Ejemplo 2.4.

Iteración	1	2	3	4	5	6	7	Grupos
1	0	0	0	1	1	1	1	$\{1,2\}$, $\{3\}$, $\{4,5,6,7\}$
2	0	0	0	0	1	1	1	$\{1,2\}$, $\{3\}$, $\{4\}$, $\{5\}$, $\{6,7\}$
3	0	1	1	0	0	1	1	$\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{5\}$, $\{6,7\}$
4	0	0	0	0	0	1	0	$\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{5\}$, $\{6\}$, $\{7\}$
5	0	0	0	0	0	0	0	$\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{5\}$, $\{6\}$, $\{7\}$

Con el fin de evitar en la medida de lo posible que las iteraciones sucesivas sean iguales (véase el ejemplo anterior para las iteraciones 4 y 5), lo que lleva a las particiones no informativas, sugerimos que el número K de pasos en el proceso jerárquico debe ser inferior al número de valores diferentes de d_e de las aristas. Este parámetro K debe ser elegido buscando un equilibrio práctico entre el número de valores diferentes de "intermediación" y el coste computacional de la obtención de todas las particiones asociadas. Por otra parte, una vez que el parámetro K se ha fijado, los distintos valores deben cumplir la condición $\alpha_1 > \cdots > \alpha_K$. Estas decisiones dependen del tamaño y las características del problema. De esta manera, vamos a considerar en este ejemplo ilustrativo $K = 4$.

Podemos construir fácilmente un dendograma a partir del orden natural inducido por el algoritmo Divide-and-Link anterior, de tal manera que se pueda obtener una agrupación jerárquica de la red. La disposición de los nodos se basa en el número binario asociado a cada clasificación. En el Ejemplo 2.4, con $K = 4$, estos números son mostrados en la Tabla 4.2. Con esta información, se puede dibujar un dendograma identificando el número de grupos y sus elementos para cada valor de α.

El valor α_0 es añadido al conjunto $\{\alpha_1, \cdots, \alpha_K\}$, el cual está asociado a la agrupación trivial sobre todo el conjunto de nodos V. Por lo tanto, en el inicio (α_0) existe un grupo único V. Luego, el proceso jerárquico divide los grupos anteriores. El proceso finalizará con α_K, produciendo tantos grupos como nodos en el grafo. En Figura 2.12a se grafica una representación de la jerarquización producida para el Ejemplo 2.4.

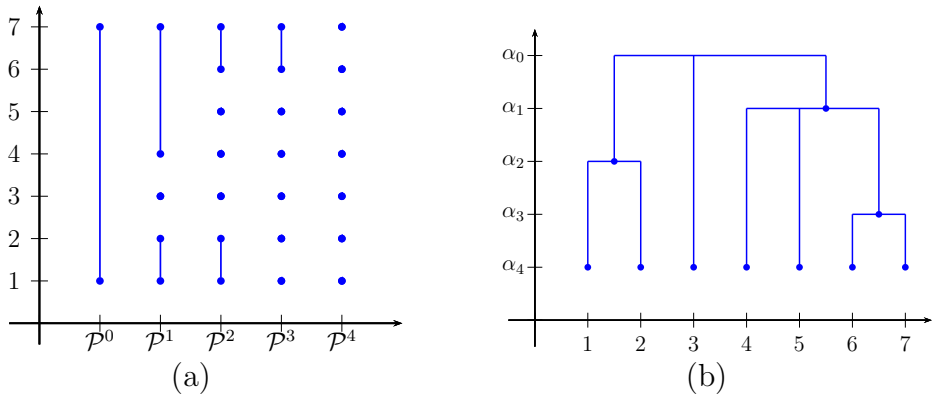


Figura 2.12: (a), Segmentación jerárquica y (b) dendograma para el Ejemplo 2.4.

Esta información puede representarse como un dendograma, el cual muestra claramente el proceso jerárquico en modo inverso: desde los nodos individuales hasta la agrupación de todos los nodos en un solo conjunto. El dendograma para el Ejemplo 2.4 es mostrado en la Figura 2.12b.

2.5. Divide-and-Link con incremento fijo

En el proceso de segmentación jerárquica de una red, en ocasiones, es de interés ver el proceso de división incrementándose en un número fijo o predeterminado de grupos en cada paso. En el proceso jerárquico bi-criterio, esto sería un caso particular en donde, para el caso de un incremento fijo, digamos ω, tomaríamos como conjunto de umbrales, los correspondientes a los valores de disimilitud que permitan ω aristas de corte efectivas

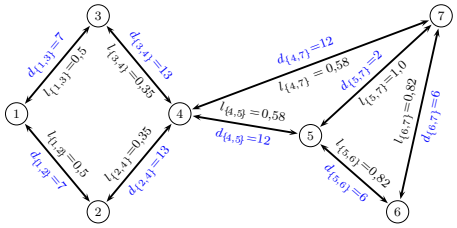
con mayor disimilitud. La ventaja de este tipo de procedimiento es que no es necesario el conocimiento a priori de un conjunto de valores $\alpha_1, \ldots, \alpha_K$, sino el valor ω : "número de grupos adicionales que se forman en cada paso" (pues los valores de los α_i quedan determinados por ω en cada iteración).

Formalmente, para generar una segmentación jerárquica con incremento ω, se utiliza un conjunto de valores de umbrales

$$\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_{K_\omega} \tag{2.16}$$

de tal forma que para cada α_i, el conjunto de aristas $T^i = E_d^i \cup E_l^i$ asociado cumple que $\left|E_d^i\right| = \omega$, donde E_d^i y E_l^i son los conjuntos de aristas de disimilitud y afinidad, respectivamente; que forman el árbol soporte en el paso i. Note que en (2.16) usamos "mayor o igual" en lugar de "mayor estricto" como se venia utilizando, debido a que puede ocurrir igualdades de valores de umbrales para garantizar la cantidad ω de incrementos.

Ejemplo 2.5 (Incremento $\omega = 1$).
En el ejemplo anterior, construyamos una segmentación jerárquica con un incremento de un grupo adicional formado en cada paso, es decir, $\omega = 1$.



Disimilitud

$$\underset{(d_{\{3,4\}})}{13} \geq \underset{(d_{\{2,4\}})}{13} \geq \underset{(d_{\{4,5\}})}{12} \geq \underset{(d_{\{4,7\}})}{12} \geq \underset{(d_{\{1,2\}})}{7} \geq \underset{(d_{\{1,3\}})}{7} \geq \underset{(d_{\{6,7\}})}{6} \geq \underset{(d_{\{5,6\}})}{6} \geq \underset{(d_{\{5,7\}})}{2}$$

Afinidad

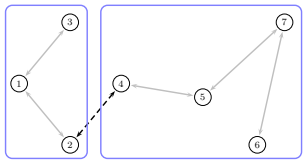
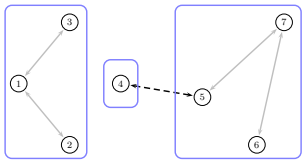
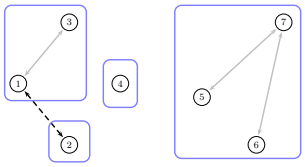
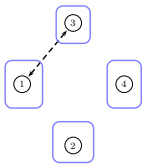
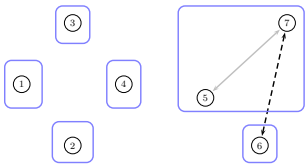
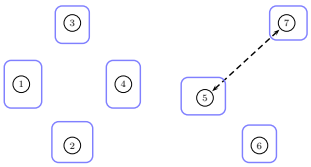
$$\underset{(l_{\{5,7\}})}{1{,}0} \geq \underset{(l_{\{6,7\}})}{0{,}82} \geq \underset{(l_{\{5,6\}})}{0{,}82} \geq \underset{(l_{\{4,5\}})}{0{,}58} \geq \underset{(l_{\{4,7\}})}{0{,}58} \geq \underset{(l_{\{1,2\}})}{0{,}5} \geq \underset{(l_{\{1,3\}})}{0{,}5} \geq \underset{(l_{\{2,4\}})}{0{,}35} \geq \underset{(l_{\{3,4\}})}{0{,}35}$$

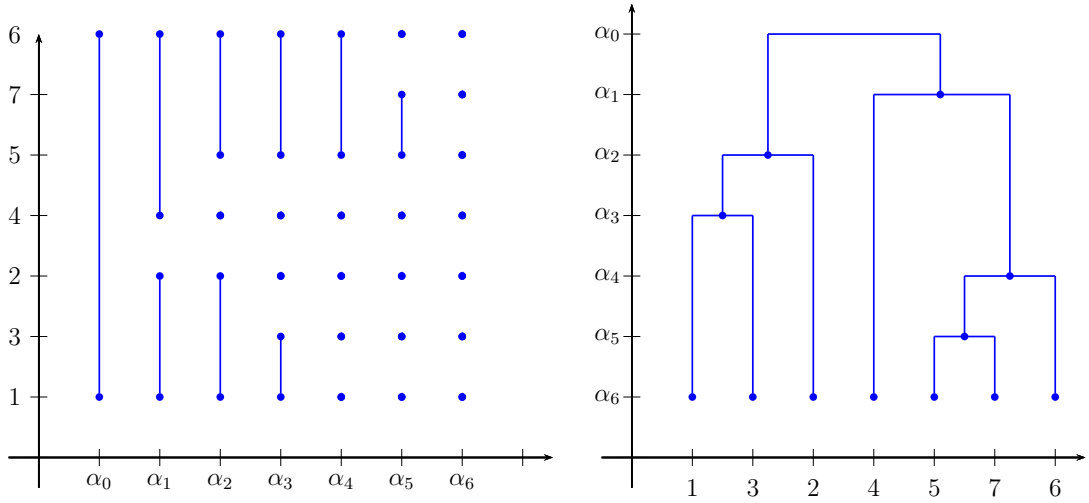
Figura 2.13: Red N: aristas en orden decreciente según disimilitud y afinidad.

Apoyados en la Figura 2.13, realizamos un proceso equivalente al de Ejemplo 2.4, garantizando que $\left|E_d^i\right| = \omega = 1$, es decir; buscamos los valores $\alpha_1 \geq \ldots \geq \alpha_K$ que cumplan que, para cada i: $\left|E_d^i\right| = 1$, donde si $e_1 \in E_d^i$ entonces $d_{e_1} \geq \alpha_i \geq d_e$ para todo $e \in E^i$. Así se obtiene la segmentación jerárquica que se muestra a continuación:

El dendograma asociado a esta partición es:

Tabla 2.4: Partición jerárquica para $\omega = 1$.

Paso 1.	Paso 2.	Paso 3.
		
		
Paso 4.	Paso 5.	Paso 6.



(a) (b)

Figura 2.14: (a) agrupación, y (b) dendograma, para el Ejemplo 2.4 con $\omega = 1$.

2.6. Anexo: Pseudocódigo del algoritmo D&L

1.- Input:

$G = (V, E)$ % the original graph

K % the number of hierarchical levels

$\{\alpha_1, \alpha_2, \ldots, \alpha_K\}$ % threshold values for the hierarchical levels

$(\overline{d} = \text{máx}_{e\in E}\{d_e\} = \alpha_1 > \alpha_2 > \cdots > \alpha_K = \underline{d} = \text{mín}_{e\in E}\{d_e\})$

2.- Inicialization:

$t = 0 \quad E^t = E \quad G^t = (G, E)$ % The initial graph

$s_0 = 1 \quad \mathcal{P}^0 = \{C_1^0\}$ % ($C_1^0 = V$, the initial partition)

3.- Process:

do t=1,K

The values of $\{d_e \mid e \in E^t\}$ are updated % (the divisive criterion)

The values of $\{l_e \mid e \in E^t\}$ are updated % (the linking criterion)

$E_f \equiv \left\{e = \{a, b\} \in E^{t-1} \mid a \in C_i^{t-1}; b \in C_j^{t-1}; i \neq j; i, j \in \{1, 2, \ldots, s_{t-1}\}\right\}$

$E^t = E^{t-1} - E_f$

$E_d = \{e_1, e_2, \ldots, e_{n_1}\} \subset E^t$ % list of divisive edges verifying:

$d_{e_1} \geq d_{e_2} \geq \cdots \geq d_{e_{n_1}} \geq \alpha_t$

$E_l = \{e_{n_1+1}, e_{n_1+2}, \ldots, e_{n_1+n_2}\} \subset E^t$ % list of linking edges verifying:

$d_{e_i} < \alpha_t \quad \forall i \in \{n_1 + 1, n_1 + 2, \ldots, n_1 + n_2\}$

$l_{e_{n_1+1}} \geq l_{e_{n_1+2}} \geq \cdots \geq l_{e_{n_1+n_2}}$

$L = E_d \cup E_l = \{e_1, e_2, \ldots, e_{n_1}, e_{n_1+1}, \ldots, e_{n_1+n_2}\}$ % List of edges

$F^t = (V, W^t) \subset G^t = (V, E^t)$ % F^t is the spanning forest of G^t, construction:

$W^t = \emptyset$

do j=1,$n_1 + n_2$

if $(V, W^t \cup e_j)$ is acyclic

$W^t = W^t \cup e_j$

endif

enddo

$\boldsymbol{\mathcal{P}}^t = \{C_1^t, C_2^t, \ldots, C_{s_t}^t\}$ is the partition of G^t, construction:

$W_l^t = W^t \cap E_l$, the divisive edges are deleted from W^t

C_i^t, for $i = 1, \ldots, s_t$ are the connected components of $H^t = (V, W_l^t)$

enddo

Capítulo 3

Redes Sociales: Detección de Comunidades

Contenido

3.1. Introducción

El análisis de redes complejas es un tema que ha despertado un interés creciente en los últimos años. En particular, el estudio de las propiedades topológicas de una red desempeña un papel importante en áreas como ciencias de la computación, biología y ciencias sociales, entre otras, (ver, por ejemplo [37, 80, 78, 106, 43, 44]). En particular, las redes sociales complejas representan un interesante campo de aplicación, y el descubrimiento de las comunidades inherentes en una red social suele ser uno de los principales objetivos a la hora de comprender mejor la red a analizar.

Al igual que en el clustering estadístico clásico, las redes pueden analizarse usando enfoques con estructuras jerárquicas y no jerárquicas. El objetivo principal de los métodos no jerárquicos, ampliamente usados en la literatura, es la obtención de una partición de un grafo de acuerdo a un cierto criterio u objetivo, produciendo como resultado una

partición de la red (ver, por ejemplo, [37, 97] donde se muestran diversos enfoques y aplicaciones).

Sin embargo, en algunas situaciones, como por ejemplo un proceso de división de una compañía o un grupo de amigos, estos procesos se llevan a cabo de una manera dinámica; debido a esto, en ocasiones es más relevante conocer todo el proceso de división en lugar de la simple imagen final (o última división). En tales casos, un enfoque de agrupación jerárquica es más apropiado debido a que el proceso general de división puede ser visualizado brindando ventajas de especificidad. Por ejemplo, como señala [23], el conocimiento de la estructura jerárquica puede ser usado para predecir la falta de conexión de una forma más realista. Por esta razón, nuestro trabajo lo hemos centrado en la segmentación jerárquica de redes. Como se subraya en [23], *la jerarquía es un principio central de organización de las redes complejas, capaces de ofrecer una idea de muchos fenómenos en la red.*

Muchos problemas reales, y en particular los problemas de redes sociales, pueden modelarse a través de teoría de grafos (usando nodos y aristas). En el caso de redes sociales, los nodos del grafo corresponden a los individuos de la red; y dos individuos están unidos por una arista si pueden ponerse en contacto entre sí directamente (por ejemplo, si viven en un mismo edificio, o pertenecen a la misma organización laboral o son miembros de un mismo foro de internet, etc).

El enfoque Divide-and-Link, introducido en el Capítulo anterior, será aplicado al análisis de algunas redes sociales que se describen en la Sección 3.3. Este capítulo termina con algunos comentarios finales que describen los principales logros con la metodología jerárquica del algoritmo Divide-and-Link (ver Sección 5.1).

3.2. Algoritmo General de Divide-and-Link

En el análisis de redes es frecuente encontrarse con "pesos" o medidas de afinidad y disimilitud, asociadas a las aristas, que dependen de la estructura propia de la componente conexa del grafo a la que pertenecen; de tal forma que si se hace una variación en el grafo, el valor respectivo puede variar. Debido a esto, cuando se hace una segmentación jerárquica de una red, es claro que de un paso a otro, la estructura de la red resultante cambia. En estos casos, es recomendable actualizar los valores de afinidad o disimilitud (o ambos), en cada paso del proceso jerárquico. Este proceso es el que hemos llamado "D&L con pesos variables".

Formalmente, considere una red $N = (G, \mathbf{D})$, para $G = (V, E)$ y conjunto de distancias asociadas a las aristas $\mathbf{D} = D \cup L$, donde $D = \{d_e \mid e \in E\}$ y $L = \{l_e \mid e \in E\}$ (cada arista tiene asociado dos valores: disimilaridad d_e, y afinidad l_e). Dados los umbrales $\alpha_0 > \overline{d} \geq \alpha_1 \geq \cdots \geq \alpha_K$, donde $\overline{d} = \text{máx}\{d_e \mid e \in E\}$; el algoritmo D&L general sigue los siguientes pasos:

Paso $i = 0$.

En este paso inicial, $E^0 = E$, luego $G^0 = (V, E^0) = (V, E) = G$; también

$$D^0 = \left\{ d_e^0 \mid e \in E^0 \right\} = D \qquad \text{y} \qquad L^0 = \left\{ l_e^0 \mid e \in E^0 \right\} = L$$

donde d_e^0 y l_e^0 son las medidas de disimilitud y afinidad para $e \in E^0$, respecto al grafo G^0. Para este instante inicial, la partición que se obtiene es $\mathcal{P}^0 = \{C_1^0\}$ con $C_1^0 = V$.

Paso i.

Supongamos que, en este paso, se ha obtenido una partición $\mathcal{P}^i = \left\{C_1^i, \ldots, C_{s_i}^i\right\}$, a partir del grafo $G^i(V, E^i)$.

Paso $i+1$.

Consideremos el grafo $G^{i+1} = (V, E^{i+1})$, donde el conjunto de aristas E^{i+1} se construye siguiendo la fórmula de recurrencia:

$$E^{i+1} = E^i \setminus E_f^i \tag{3.1}$$

donde E^i es el conjunto de aristas del grafo G^i en el paso i, y

$$E_f^i = \left\{ e = \{a, b\} \in E^i \;\middle|\; a \in C_q^i,\ b \in C_r^i \;\ ;\quad C_q^i, C_r^i \in \mathcal{P}^i \ ,\ q \neq r, \right\} \tag{3.2}$$

para $\mathcal{P}^i$ la partición formada en el paso i. Note que E_f^i es el conjunto de aristas que conecta grupos en la partición $\mathcal{P}^i$. Como el grafo G^{i+1} queda complemente determinado por (3.1), podemos actualizar, para cada arista $e \in E^{i+1}$, los valores d_e^{i+1} y l_e^{i+1} que dependen del grafo G^{i+1}; y así obtener los conjuntos:

$$D^{i+1} = \left\{ \ d_e^{i+1} \ \middle|\ e \in E^{i+1} \right\} \qquad \text{y} \qquad L^{i+1} = \left\{ \ l_e^{i+1} \ \middle|\ e \in E^{i+1} \right\}.$$

Ahora bien, determinados G^{i+1}, D^{i+1} y L^{i+1}, construimos el grafo parcial $F^{i+1} = (V, T^{i+1})$ de acuerdo a las siguientes reglas:

(I) Siguiendo un proceso similar al de Kruskal, se van incorporando a T^{i+1} aristas (de corte), en orden decreciente, del conjunto

$$E_d^{i+1} = \left\{ e \in E^{i+1} \quad \middle| \quad d_e^{i+1} \geq \alpha_{i+1} \right\} \tag{3.3}$$

con la condición de que la arista a adicionar no forme ciclos.

(II) Si en el paso anterior, las aristas en T^{i+1} no constituyen árbol soporte para G^{i+1} (o bosque soporte, si G^{i+1} no es conexo), entonces se continua adicionando a T^{i+1}, y en orden decreciente, aristas pertenecientes al conjunto

$$E_l^{i+1} = \left\{ e \in E^{i+1} \setminus E_d^{i+1} \quad \middle| \quad l_e^{i+1} \in L^{i+1} \right\} \tag{3.4}$$

con la misma condición anterior de no formar ciclos. Note que las aristas que escogen del conjunto E_l^{i+1} son las que tienen una afinidad mayor.

El proceso anterior termina cuando:

1. F^{i+1} sea un árbol soporte para G^{i+1}, o
2. F^{i+1} sea un bosque soporte para G^{i+1}, con G^{i+1} no conexo, o
3. cuando se agoten las aristas de E_d^{i+1} y E_l^{i+1}.

Finalmente, la partición $i+1$ es: $\mathcal{P}^{i+1} = \left\{C_1^{i+1}, C_2^{i+1}, \ldots, C_{s_{i+1}}^{i+1}\right\}$, donde los $C_j^{i+1} \subset V$, con $j = 1, 2, \ldots, s_{i+1}$, son las componentes conexas del grafo $\left(V\ ,\ T^{i+1} \setminus E_d^{i+1}\right)$.

Teorema 3.1: *La secuencia* $\left\{\mathcal{P}^i\right\}_{i=0}^{K}$ *de particiones definida por el algoritmo D&L es jerárquica.*

PRUEBA: Para demostrar el teorema basta probar que para cualquier $C_j^{i+1} \in \mathcal{P}^{i+1}$, existe un $C_m^i \in \mathcal{P}^i$ tal que $C_j^{i+1} \subseteq C_m^i$ $\left(\text{i.e., } C_j^{i+1} \cap C_m^i = C_j^{i+1}\right)$. Razonemos por reducción al absurdo:

Sin pérdida de generalidad, supongamos que C_j^{i+1} está contenido en dos conjuntos distintos de $\mathcal{P}^i$. Es decir, $\exists m, n;\ m \neq n$ tal que $C_j^{i+1} \subseteq C_m^i \cup C_n^i$ para $C_m^i, C_n^i \in \mathcal{P}^i$. Luego, $C_j^{i+1} \cap C_m^i \neq \emptyset$ y $C_j^{i+1} \cap C_n^i \neq \emptyset$ y son conjuntos disjuntos (pues C_m^i y C_n^i son componentes conexas del grafo $F^i = (V, T^i \setminus E_d^i)$, i.e., $C_m^i \cap C_n^i = \emptyset$, $m \neq n$).
Sean $a, b \in V$ tales que $a \in C_j^{i+1} \cap C_m^i$ y $b \in C_j^{i+1} \cap C_n^i$; entonces $a, b \in C_j^{i+1}$, y como C_j^{i+1} es conexo, existe un camino entre a y b:

$$\left\langle a,\ e_{a,z_1},\ z_1,\ e_{z_1,z_2},\ z_2,\ \ldots,\ z_\theta,\ e_{z_\theta,b},\ b\right\rangle \tag{3.5}$$

con los nodos $\{a, z_1, \ldots, z_\theta, b\} \subset C_j^{i+1}$ y las aristas $\{e_{a,z_1}, e_{z_1,z_2} \ldots, e_{z_\theta,b}\} \subset T^{i+1} \setminus E_d^{i+1} \subset E^{i+1} \subset E^i$. Ahora bien, dado que $a \in C_m^i$ y $b \in C_n^i$ entonces existen un par de nodos consecutivos en el camino (3.5), digamos $z_r, z_{r+1} \in \{a, z_1, \ldots, z_\theta, b\}$, cumpliendo que $z_r \in C_m^i$ y $z_{r+1} \in C_n^i$. Así, existe en el camino (3.5) una arista $e_{z_r,z_{r+1}}$ conectando dichos nodos. En resumen, se tiene que $e_{z_r,z_{r+1}} \in E^i$ y tal que $z_r \in C_m^i$ y $z_{r+1} \in C_n^i$, para $C_m^i, C_n^i \in \mathcal{P}^i$, con $m \neq n$, así: $e_{z_r,z_{r+1}} \in E_j^i$. Concluimos que $e_{z_r,z_{r+1}} \notin E^{i+1} = E^i - E_j^i$. Absurdo!, luego $m = n$. ∎

Observación:

Si en el algoritmo D&L se tiene la condición de un incremento fijo, digamos ω, entonces el procedimiento anterior se realiza de manera equivalente, salvo que, para cada i, se toma el valor α_i que garantice que conjunto

$$E_d^i = \left\{ e_1, \ldots, e_\omega \in E^i \;\middle|\; d_{e_1}^i \geq \cdots \geq d_{e_\omega}^i \geq \alpha_i \geq d_e^i\ ; \quad \forall e \in E^i \setminus \{e_1, \ldots, e_\omega\} \right\} \tag{3.6}$$

contenga exactamente ω aristas de corte que no formen ciclos. En el algoritmo D&L, para un incremento fijo, basta con utilizar la definición de E_d^i dada en (3.6) en lugar de (3.3).

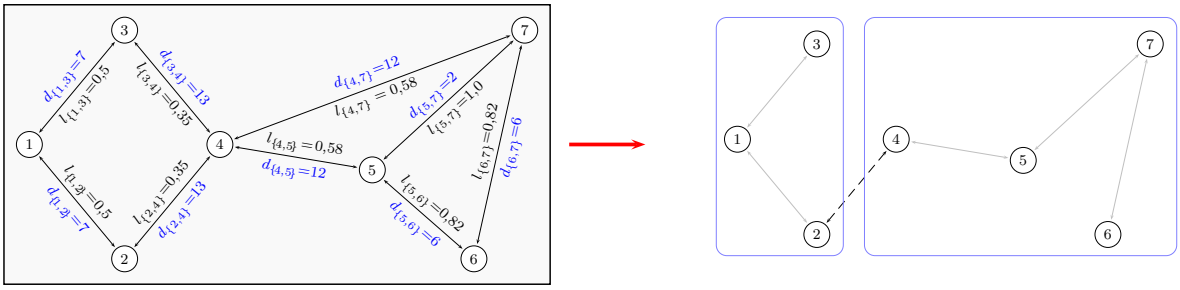
Ejemplo 3.1 (D&L con incremento $\omega = 1$ y disimilitud variable).
Continuemos con el Ejemplo 2.4, pero en este caso haremos una segmentación jerárquica usando un incremento $\omega = 1$ y un peso de disimilitud variable para las aristas (conservando constante el peso asociado a la afinidad). Más específicamente, usaremos como medida de disimilitud la intermediación o "betweenness SP" asociada a la arista (introducida en la Sección 1.3), y actualizada luego de cada iteración (note que, en cada paso, la disimilaridad usando "betweenness", será calculada dependiendo del grafo resultante; sin embargo, la afinidad asociada a cada arista será constante desde el principio).

Paso 1:

Disimilitud: $\overset{\checkmark}{\underset{(d_{\{2,4\}})}{13}} \geq \underset{(d_{\{3,4\}})}{13} \geq \underset{(d_{\{4,5\}})}{12} \geq \underset{(d_{\{4,7\}})}{12} \geq \underset{(d_{\{1,2\}})}{7} \geq \underset{(d_{\{1,3\}})}{7} \geq \underset{(d_{\{6,7\}})}{6} \geq \underset{(d_{\{5,6\}})}{6} \geq \underset{(d_{\{5,7\}})}{2}$

Afinidad: $\underset{(l_{\{5,7\}})}{\overset{\checkmark}{1{,}0}} \geq \underset{(l_{\{6,7\}})}{\overset{\checkmark}{0{,}82}} \geq \underset{(l_{\{5,6\}})}{0{,}82} \geq \underset{(l_{\{4,5\}})}{\overset{\checkmark}{0{,}58}} \geq \underset{(l_{\{4,7\}})}{0{,}58} \geq \underset{(l_{\{1,2\}})}{\overset{\checkmark}{0{,}5}} \geq \underset{(l_{\{1,3\}})}{\overset{\checkmark}{0{,}5}} \geq \underset{(l_{\{2,4\}})}{0{,}35} \geq \underset{(l_{\{3,4\}})}{0{,}35}$

En la Figura 3.1b, se muestra la partición $\mathcal{P}^1 = \{C_1^1, C_2^1\}$, donde $C_1^1 = \{1,2,3\}$ y $C_2^1 = \{4,5,6,7\}$.



(a) (b)

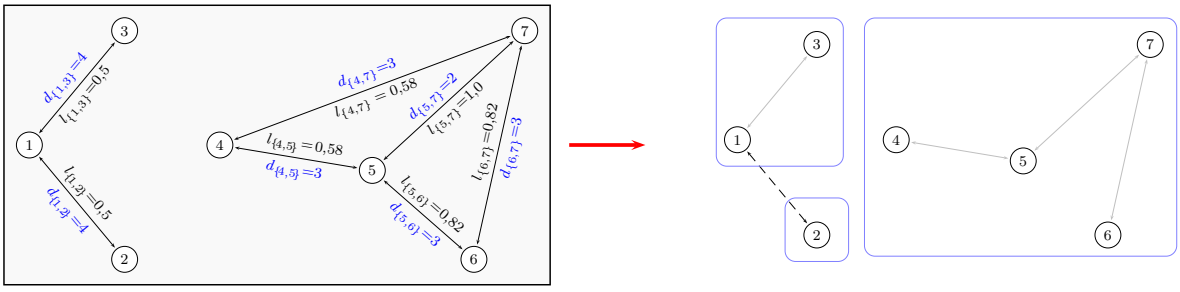
Figura 3.1: (a). Grafo $G^1 = (V, E^1)$, (b). Árbol $F^1 = (V, T^1)$

Paso 2:

Disimilitud: $\underset{(d_{\{1,2\}})}{\overset{\checkmark}{4}} \geq \underset{(d_{\{1,3\}})}{4} \geq \underset{(d_{\{4,5\}})}{3} \geq \underset{(d_{\{4,7\}})}{3} \geq \underset{(d_{\{6,7\}})}{3} \geq \underset{(d_{\{5,6\}})}{3} \geq \underset{(d_{\{5,7\}})}{2}$ (nueva)

Afinidad: $\underset{(l_{\{5,7\}})}{\overset{\checkmark}{1{,}0}} \geq \underset{(l_{\{6,7\}})}{\overset{\checkmark}{0{,}82}} \geq \underset{(l_{\{5,6\}})}{0{,}82} \geq \underset{(l_{\{4,5\}})}{\overset{\checkmark}{0{,}58}} \geq \underset{(l_{\{4,7\}})}{0{,}58} \geq \underset{(l_{\{1,2\}})}{0{,}5} \geq \underset{(l_{\{1,3\}})}{\overset{\checkmark}{0{,}5}}$

En la Figura 3.2b, se muestra la partición $\mathcal{P}^2 = \{C_1^2, C_2^2, C_3^2\}$, donde $C_1^2 = \{1,3\}$, $C_2^2 = \{2\}$ y $C_3^2 = \{4,5,6,7\}$.



(a) (b)

Figura 3.2: (a). Grafo $G^2 = (V, E^2)$, (b). Bosque $F^2 = (V, T^2)$

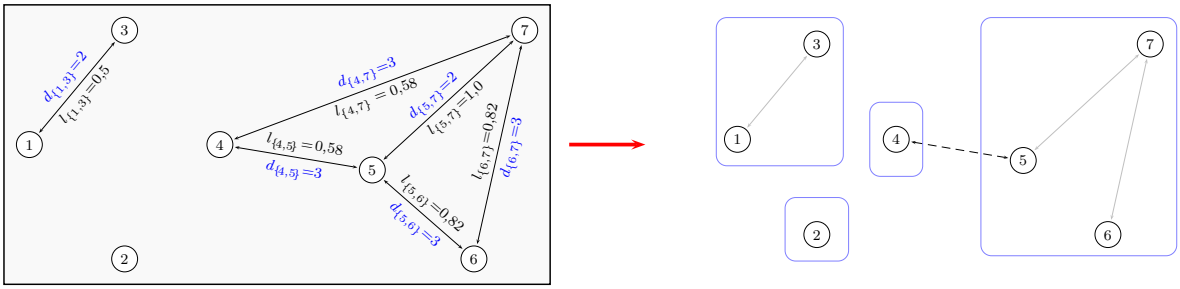
Paso 3:

Disimilitud: $\underset{(d_{\{4,5\}})}{\overset{\checkmark}{3}} \geq \underset{(d_{\{4,7\}})}{3} \geq \underset{(d_{\{6,7\}})}{3} \geq \underset{(d_{\{5,6\}})}{3} \geq \underset{(d_{\{1,3\}})}{2} \geq \underset{(d_{\{5,7\}})}{2}$ (nueva)

Afinidad: $\underset{(l_{\{5,7\}})}{\overset{\checkmark}{1{,}0}} \geq \underset{(l_{\{6,7\}})}{\overset{\checkmark}{0{,}82}} \geq \underset{(l_{\{5,6\}})}{0{,}82} \geq \underset{(l_{\{4,5\}})}{0{,}58} \geq \underset{(l_{\{4,7\}})}{0{,}58} \geq \underset{(l_{\{1,3\}})}{\overset{\checkmark}{0{,}5}}$

En la Figura 3.3b, se muestra la partición $\mathcal{P}^3 = \{C_1^3, C_2^3, C_3^3, C_4^3\}$, donde $C_1^3 = \{1,3\}$, $C_2^3 = \{2\}$, $C_3^3 = \{4\}$ y $C_4^3 = \{5,6,7\}$.

Paso 4:



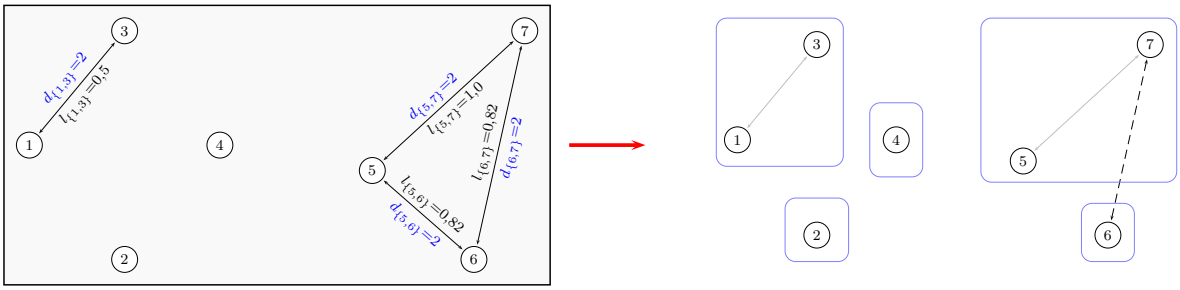
(a) (b)

Figura 3.3: (a). Grafo $G^3 = (V, E^3)$, (b). Bosque $F^3 = (V, T^3)$

Disimilitud: $\underset{(d_{\{6,7\}})}{\overset{\checkmark}{2}} \geq \underset{(d_{\{5,6\}})}{2} \geq \underset{(d_{\{1,3\}})}{2} \geq \underset{(d_{\{5,7\}})}{2}$ (nueva)

Afinidad: $\underset{(l_{\{5,7\}})}{\overset{\checkmark}{1{,}0}} \geq \underset{(l_{\{6,7\}})}{\overset{\checkmark}{0{,}82}} \geq \underset{(l_{\{5,6\}})}{0{,}82} \geq \underset{(l_{\{1,3\}})}{0{,}5}$

En la Figura 3.4b, se muestra la partición $\mathcal{P}^4 = \{C_1^4, C_2^4, C_3^4, C_4^4, C_5^4\}$, donde $C_1^4 = \{1, 3\}$, $C_2^4 = \{2\}$, $C_3^4 = \{4\}$, $C_4^4 = \{6\}$ y $C_5^4 = \{5, 7\}$.



(a) (b)

Figura 3.4: (a). Grafo $G^4 = (V, E^4)$, (b). Bosque $F^4 = (V, T^4)$

Paso 5:

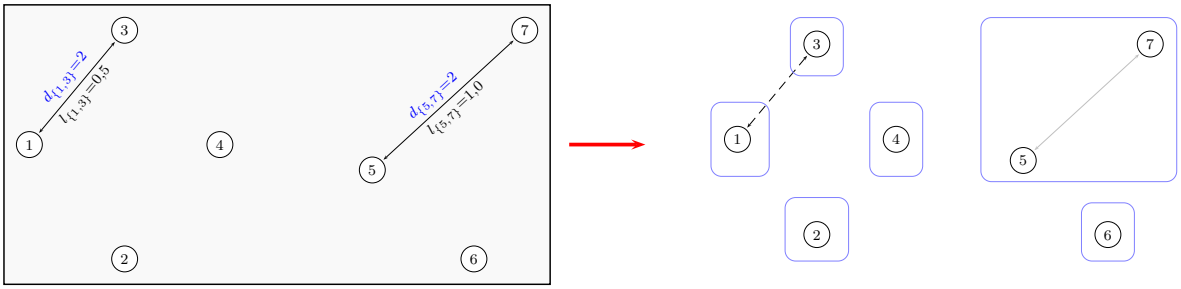
Disimilitud: $\underset{(d_{\{1,3\}})}{\overset{\checkmark}{2}} \geq \underset{(d_{\{5,7\}})}{2}$ (nueva) Afinidad: $\underset{(l_{\{5,7\}})}{\overset{\checkmark}{1{,}0}} \geq \underset{(l_{\{1,3\}})}{0{,}5}$

En la Figura 3.5b, se muestra la partición $\mathcal{P}^5 = \{C_1^5, C_2^5, C_3^5, C_4^5, C_5^5, C_6^5\}$, donde $C_1^5 = \{1\}$, $C_2^5 = \{3\}$, $C_3^5 = \{2\}$, $C_4^5 = \{4\}$, $C_5^5 = \{6\}$ y $C_6^5 = \{5, 7\}$.

Paso 6:

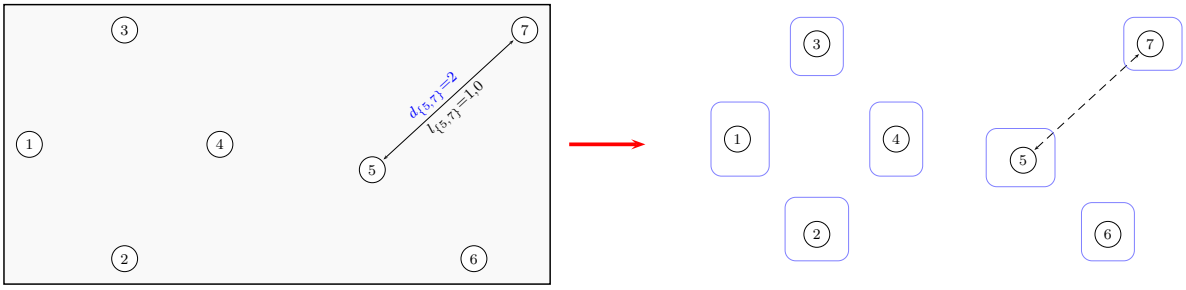
Disimilitud: $\underset{(d_{\{5,7\}})}{\overset{\checkmark}{2}}$ (nueva) Afinidad: $\underset{(l_{\{5,7\}})}{1{,}0}$

En la Figura 3.6b, se muestra la partición $\mathcal{P}^6 = \{C_1^6, C_2^6, C_3^6, C_4^6, C_5^6, C_6^6, C_7^6\}$, donde $C_1^6 = \{1\}$, $C_2^6 = \{3\}$, $C_3^6 = \{2\}$, $C_4^6 = \{4\}$, $C_5^6 = \{6\}$, $C_6^6 = \{5\}$ y $C_7^6 = \{7\}$.



(a) (b)

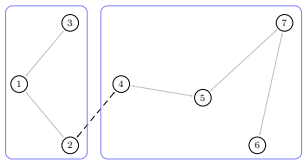
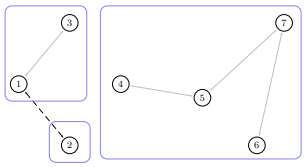
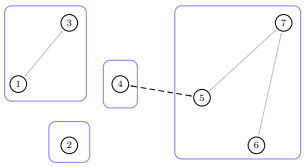
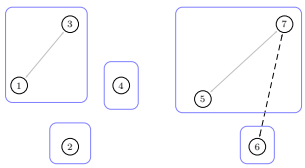
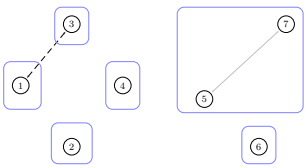
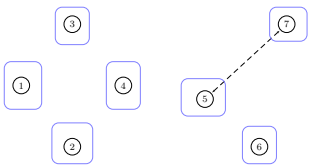
Figura 3.5: (a). Grafo $G^5 = (V, E^5)$, (b). Bosque $F^5 = (V, T^5)$



(a) (b)

Figura 3.6: (a). Grafo $G^6 = (V, E^6)$, (b). Bosque $F^6 = (V, T^6)$

Tabla 3.1: Segmentación jerárquica para $\omega = 1$ y disimilitud variable.

Paso 1.	Paso 2.	Paso 3.
		
		
Paso 4.	Paso 5.	Paso 6.

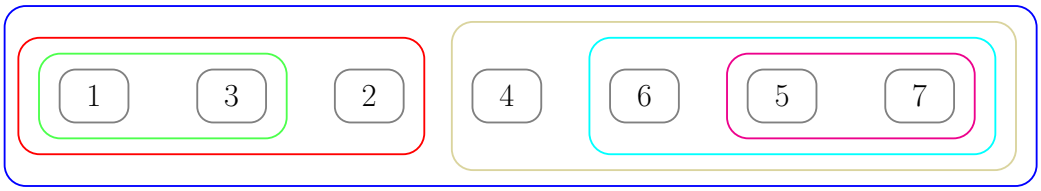
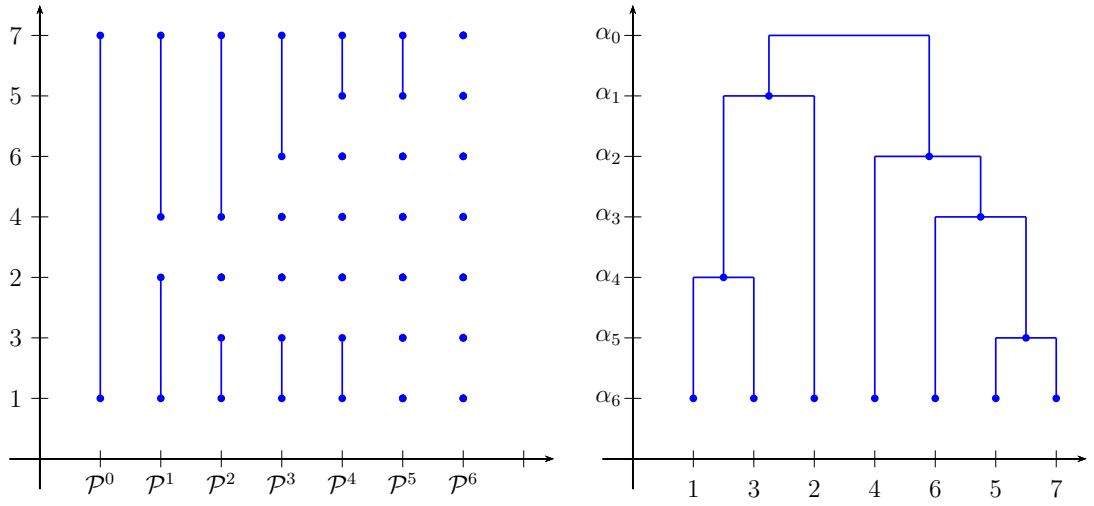


Figura 3.7: Diagrama de jerarquización producida por D&L



(a) (b)

Figura 3.8: (a) agrupación, y (b) dendograma, para el Ejemplo 2.4 con $\omega = 1$ y disimilitud variable.

3.2.1. Implementación del algoritmo D&L

A continuación mostramos el procedimiento que se debe llevar a cabo para implementar el algoritmo D&L general propuesto anteriormente.

Paso I. *Entradas:*

- `G=(V,E)` (grafo inicial)
- `ALP` (vector de umbrales `ALP`$= \{\alpha_1, \ldots, \alpha_K\}$, de dimensión `K`).
- `d(e,G); l(e,G)` (funciones de disimilitud `d` y afinidad `l` para la arista `e` en el grafo `G`).

Paso II. *Inicializar:* partición trivial.

- `E0=E`;
- `P0={V}` (Si el grafo inicial `G` no es conexo, entonces `P0` contendrá a todas sus componentes conexas, i.e, `P0`$=\{$`V`$_1, \ldots,$ `V`$_s\}$).
- `h=1`;

Paso III. *Segmentación jerárquica.*

- `alp=ALP(h); Ef=[ ]; El=[ ]; Ed=[ ];`
- Construir

$$\texttt{Ef} := \left\{ \texttt{e} = \{a, b\} \in \texttt{E0} \;\middle|\; a \in \texttt{A},\ b \in \texttt{B};\quad \texttt{A} \neq \texttt{B},\ \texttt{A,B} \in \texttt{P0} \right\}$$

- E1=E0\Ef; G1=(V,E1);
- Construir:
  $$\texttt{Ed}=\Big\{\texttt{e}_1,\ldots,\texttt{e}_{n_1} \;\Big|\; \texttt{d(e}_1\texttt{,G1)} \geq \cdots \geq \texttt{d(e}_{n_1}\texttt{,G1)} \geq \texttt{alp}\Big\}$$
  $$\texttt{El}=\Big\{\texttt{e}_{n_1+1},\ldots,\texttt{e}_{n_2} \;\Big|\; \texttt{l(e}_{n_1+1}\texttt{,G1)} \geq \cdots \geq \texttt{l(e}_{n_2}\texttt{,G1)}\Big\}$$
- i=1; j= n_1+1; T=[ ];

Paso IV. *Bosque soporte:* aristas de disimilitud.

- Si para $\texttt{e}_i \in \texttt{Ed}$, el grafo $\texttt{F=(V,T} \cup \{\texttt{e}_i\})$ es acíclico; hacer $\texttt{T=T} \cup \{\texttt{e}_i\}$.
- i=i+1;
- Si i $\leq n_1$; ir al **Paso IV**; en caso contrario, ir al **Paso V**.

Paso V. *Bosque soporte:* aristas de afinidad.

- Si para $\texttt{e}_j \in \texttt{El}$, el grafo $\texttt{F=(V,T} \cup \{\texttt{e}_j\})$ es acíclico; hacer $\texttt{T=T} \cup \{\texttt{e}_j\}$.
- j=j+1;
- Si j $\leq n_2$; ir al **Paso V**; en caso contrario, ir al **Paso VI**.

Paso VI. *Agrupaciones:*

- Almacenar $\texttt{P1} = \{\texttt{C}_0,\ldots,\texttt{C}_s\}$ donde los $\texttt{C}_i$ son las componentes conexas del grafo $\big(\texttt{V,T} \setminus \texttt{Ed}\big)$.
- h=h+1;
- Si h $\leq$ K, hacer E0=E1; P0=P1; e ir al **Paso III**; en caso contrario, **Fin**.

3.3. D&L en Detección de Comunidades

En esta sección mostraremos cómo adaptar el modelo de "Divide and Link" propuesto en el Capítulo 2, para el problema de detección de comunidades en redes, teniendo en cuenta los dos criterios fundamentales supuestos para "romper" y "agrupar" los individuos de la red (o nodos del grafo). El criterio que será usado en esta ocasión para dividir el grafo será a través de la medida de centralidad por *intermediación* para aristas propuesta en [43] por Girvan y Newman como una adaptación para aristas de la centralidad clásica para nodos por intermediación. (otras medidas pudieran ser usadas para reducir la complejidad computacional). Formalmente, la medida de intermediación dada en [43] se define como:

$$d_e = d_{(i,j)} = w_{ij}^B = \sum_{\substack{(r,s)\\ r\neq s}} I_{rs}\Big(\,(ij)\,\Big), \tag{3.7}$$

donde $I_{rs}\big(\,(ij)\,\big) = 1$ si el camino más corto de r a s usa el enlace (i,j). Si existe más de un "camino corto" entre r y s, $I_{rs}\big(\,(ij)\,\big)$ se divide por la cantidad de caminos. Para una arista, la medida de intermediación dada en (3.7) representa el número de

comunicaciones entre pares de nodos de la red que la "necesitan" si la comunicación entre cada par de nodos se hace a través del *camino más corto*.

Otro parámetro que se debe determinar en el algoritmo "Divide and Link" que usaremos es la secuencia de valores alfa (umbrales). Esta secuencia nos dará para cada iteración un conjunto de enlaces que serán usados para dividir la red (romper el grafo). Teniendo en cuenta que queremos mostrar las diferencias con otros algoritmos jerárquicos, buscamos una secuencia de alfa-valores que garanticen que la partición en la iteración t tenga un grupo más de elementos que la partición en la iteración anterior (es decir, $|\mathcal{P}^t| = |\mathcal{P}^{t-1}| + 1$); esto se logra aplicando el algoritmo D&L con incremento fijo $\omega = 1$. Es importante recalcar que nuestro algoritmo, para una iteración dada, no necesariamente rompe un grupo conectado en dos grupos. Si se tiene más de un enlace para "romper" la red, el grupo puede romperse, en una iteración, en varios subgrupos.

Por último, para poder adaptar el criterio Divide-and-Link a la detección de comunidades, debemos determinar un criterio de enlace.

El criterio de enlace que hemos usado es una medida de similitud (o afinidad) entre nodos definida en [60] que muestra muy buenos resultados. Sean i y j dos nodos en el grafo $G = (V, E)$, entonces la medida de similaridad entre i and j se define como

$$l_{(i,j)} = \begin{cases} \dfrac{\left|N_k(i) \cap N_k(j)\right| + 1}{\sqrt{\left|N_k(i)\right| \left|N_k(j)\right|}} & (i,j) \in E \\ 0 & \text{en otro caso} \end{cases} \tag{3.8}$$

donde $N_k(i) = \left\{ j \in V \setminus \{i\} \quad | \quad \text{dis}\,(i,j) \leq k \right\}$, para cualquier función de distancia "dis". Para este trabajo hemos asumido $k = 1$ y hemos tomado como "dis" el número mínimo de aristas necesarias para ir de i a j.

3.4. Complejidad

Como ya se ha analizado en el Capítulo 2, la complejidad computacional de nuestro algoritmo jerárquico Divide-and-Link (D&L) es $O = (df)$. Siguiendo a Fortunato en [37], la intermediación (betweennees) de todas las aristas de un grafo se pueden calcular como $O(mn)$ (o $O(n^2)$ en grafos dispersos, es decir, cuando m es cercano de n). La similitud l_e definida en [60] para $k = 1, 2$ se puede calcular con un orden $O(m)$, por lo que es evidente que, si utilizamos estas dos medidas, el orden de f está limitado por el cálculo de d_e para todas las aristas, esto es: $O(mn)$ ó $O(n^2)$ en redes dispersas. Por lo tanto, la complejidad computacional del algoritmo Divide-and-Link sobre la base de estas dos métricas es $O(dmn)$ ó $O(dn^2)$ en las redes dispersas. Sin embargo, como ya se ha señalado, podemos reducir esta complejidad (orden $O(n^3)$ si queremos mostrar todo el dendograma), teniendo en cuenta medidas alternativas, o versiones más rápidas de la medida de intermediación (que denotaremos por betweennees SP, o simplemente, BSP).

Los algoritmos que usan BSP presentan problemas computacionales, como se pone de manifiesto, por ejemplo, en [37], uno de los principales inconvenientes del *algoritmo-GN*, introducido en la Sección 1.3, es su complejidad computacional. El orden asociado con el *algoritmo-GN* es $O = (m^2 n)$ ($O(n^3)$ en redes dispersas), esto se debe a que tenemos que repetir m veces el cálculo de la BSP, la cual requiere $O(nm)$. En la práctica,

este hecho hace imposible analizar redes muy grandes, a pesar de que se puede aplicar a redes de tamaño pequeño o medio (es decir, 10^6 nodos). Con el fin de ser competitivos en redes de gran tamaño, algunos autores (ver por ejemplo, [78, 98, 108]) han propuesto técnicas más rápidas para los cálculos de la medida BSP para todas las aristas del grafo. La complejidad computacional del cálculo de la intermediación (con orden $O(nm)$) también se puede reducir drásticamente cuando se restringe a algunos nodos claves de la red. Por ejemplo, usando una amplitud inicial de búsqueda, el cálculo de la BSP tiene una complejidad lineal ($O(m)$) en grafos sin pesos, si se restringe con el camino mas corto, desde un centro para el resto de los nodos. Si solamente consideramos $f(n) < n$ nodos de la red, entonces el cálculo de la intermediación para todas las aristas podría reducirse de $O(nm)$ hasta $O(f(n)m)$. Si por ejemplo, $f(n) = \log(n)$, entonces la complejidad computacional para esta aproximación de la BSP es cercana a $O(m)$, de tal manera que el *algortimo-GN* con esta modificación podría ser reducirse a $O(m^2)$ (o también $O(n^2)$ en redes dispersas).

Tabla 3.2: Complejidad computacional de los algoritmos Girman-Newman, Moore, Walktrap, Radicchi, Newman 2012, y el algoritmo Divide-and-Link usando diferentes expresiones para d_e y l_e con actualización y sin actualización de la información.

Algoritmo	**Complejidad computacional**	**Orden en redes dispersas**
Givan Newman	$O(m^2n)$	$O(n^3)$
CNM	$O(md\log(n)) = O(mn\log(n))$	$O(n^2\log(n)) \sim O(n^2)$
Radicchi	$O(m^4/n^2)$	$O(n^2)$
Walktrap	$O(mn^2)$	$O(n^3)$
N2012	$O(n^2\log(n))$	$O(n^2\log(n)) \sim O(n^2)$
D&L con d_e=Betweeness	$O(dmn) = O(mn^2)$	$O(n^3)$
D&LF d_e=Fast Betweeness	$O(dm\log(n))$	$O(n^2)$
D&L d_e=medida Radhicchi	$O(dm^3/n^2)$	$O(dn)$
D&L $d_e = 1/l_e$	$O(dm\log(m))$	$O(n^2)$
No actualizando d_e, l_e	**Complejidad computacional**	**Orden redes dispersas**
D&L con d_e=Betweenness	$O(mn) = O(mn)$	$O(n^2)$
D&LF d_e=Fast Betweenness	$O(m\log(n))$	$O(n)$
D&L d_e=medida de Radhicchi	$O(f(m)) = O(m^3/n^2)$	$O(n)$
D&L $d_e = 1/l_e$	$O(m\log(m))$	$O(n)$

Siguiendo a Tyler y Wikinson en [98, 108], proponemos una versión rápida del cálculo de la intermediación (BSP), y que en el algoritmo Divide-and-Link denotamos como **algoritmo D&LF** (Divide-and-Link Fast). Si sólo tenemos en cuenta el $\log(n)$ de nodos con máximo grado de la red para aproximar la medida de intermediación, la complejidad computacional asociada a la *algoritmo-D&L* puede reducirse de $O(dmn)$ hasta $O(dm\log(n))$ (cercano al orden lineal en m si sólo estamos interesados en las primeras iteraciones, o al orden cuadrático si construimos el dendograma completo en redes dispersas).

En la Tabla 3.2 se muestra la complejidad computacional asociada a algunos de los algoritmos más conocidos para la detección de comunidades que tratan de resolver el problema de una manera jerárquica, así como las diferentes posibilidades del algoritmo Divide-and-Link , *algortimo-D&L* y *algortimo-D&LF*, usando alternativas para las medidas de d_e y l_e. También debemos observar que en cada iteración actualizamos los valores de d_e y l_e. Podemos tratar de reducir la complejidad computacional del *algortimo-D&L* si no re realiza dicha actualización, pero, en algunos problemas, no pueden asegurarse mejores resultados.

3.5. Resultados Computacionales

Para probar la eficacia de un algoritmo (y la detección de la comunidades no es una situación diferente), es necesario responder (o decidir) varias cuestiones. La primera es relativa a los algoritmos con los que se van a comparar los resultados. La segunda, es la selección de los ejemplos en los que se van a chequear y poner a prueba la eficacia. Finalmente, la tercera, es la medida de precisión que muestre la calidad de los resultados con los diferentes algoritmos. A continuación, vamos a tratar de responder estas preguntas con el fin de validar el rendimiento de nuestro algoritmo.

Inicialmente, vamos a comparar el algoritmo Divide-and-Link con el algoritmo de Girvan y Newman, el algoritmo CNM, el algoritmo de Walktrak [88] y el algoritmo dado por Newman en [77]. Otros algoritmos para detección de comunidades no han sido incluidos en el análisis por diferentes razones: algunos de ellos (como el expuesto por Doneti en [26], entre otros) son muy complicados de implementar puesto que el código no está disponible y sus algoritmos no son automáticos (dependen de muchos parámetros que deben ser determinados para lograr un buen desempeño); otros algoritmos (como el bien conocido algoritmo de Blondel [6]) no realizan una partición jerárquica incrementando el número de comunidades en cada iteración en una unidad, así que no podemos comparar el proceso dinámico de una manera justa.

La segunda cuestión se relaciona con la selección de ejemplos tipo test. En el contexto de la detección de comunidades, los ejemplos provenientes de una simulación cuando ellos son generados con una estructura previa la cual debe ser detectada por los algoritmos; sin embargo, ese no es nuestro caso, pues nuestro objetivo es el análisis jerárquico de una red social y no existe una "mejor" estructura a priori para validar la secuencia óptima de particiones obtenida por el algoritmo "Divide-and-Link". Por todo lo anterior, los ejemplos de prueba que hemos utilizado son los bien conocidos ejemplos en problemas de detección de comunidades que admitan una interpretación jerárquica, tales como "Karate Club Network", "The Dolphin Network", "Les Miserables Network" y "Author's Network".

Teniendo en cuenta las consideraciones anteriores, analizaremos el desempeño de nuestro algoritmo "Divide-and-Link" en las siguientes redes:

1. *Karate Club*
2. *Les Miserables*
3. *Centrality Authors* , y
4. *Dolphins*

Por último, la tercera pregunta que tenemos que considerar es relativa a las medidas de precisión que permite comparar los resultados obtenidos por algoritmos diferentes. Además de algunas características inherentes a un algoritmo cualquiera (como la complejidad computacional, o la dificultad del algoritmo para ser implementado), tenemos aquí dos problemas: el primero es relativo a la manera de comparar dos particiones de un grafo; una vez que este primera problema se resuelve, el segundo problema es cómo comparar dos secuencias de particiones.

La medida que hemos usado para representar la calidad de una partición es la "función de modularidad" (ver [43, 80] para más detalles) definida en (1.1). Dada una partición $\mathcal{P}$ de una red, la representación matricial de la modularidad se define como

$$Q_{\mathcal{P}} = \frac{1}{2m}\mathrm{Tr}\Big(\mathbf{S^T B S}\Big)$$

donde $\mathbf{B}$ es la matriz de modularidad.

Observación 3.1.
Notemos que el problema de la comparación de dos segmentaciones jerárquicas $\boldsymbol{\mathcal{D}}_1$, $\boldsymbol{\mathcal{D}}_2$ sigue sin resolverse a pesar de medidas como la modularidad, ya que la agrupación jerárquica muestra de una forma dinámica cómo se rompen los grupos. Comparar dos dendogramas no es una tarea fácil, como se muestra en el siguiente ejemplo.

Ejemplo 3.2.
Consideremos una cadena $Ch = (V, E)$ de 12 nodos ordenados de forma natural (véase la figura 3.9). Se puede demostrar que la modularidad máxima para este grafo se alcanza con una partición del grafo en tres cortes, $C_1 = \{1, 2, 3, 4\}$, $C_2 = \{5, 6, 7, 8\}$ y $C_3 = \{9, 10, 11, 12\}$.

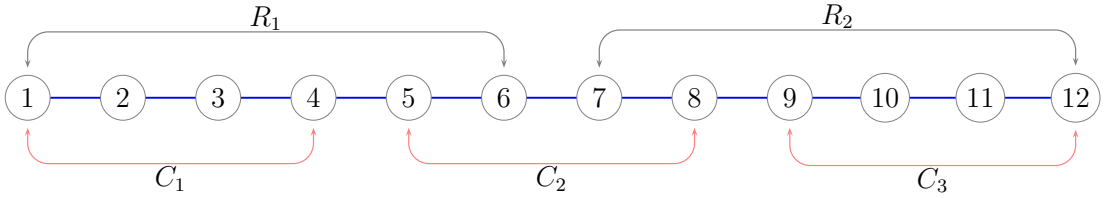


Figura 3.9: Cadena $Ch = (V, E)$ de 12 nodos

No obstante, desde un punto de vista de un algoritmo divisivo, la primera partición debe ser $R_1 = \{1, 2, 3, 4, 5, 6\}$ y $R_2 = \{7, 8, 9, 10, 11, 12\}$, que produce la partición de máxima modularidad para dos grupos. Tenga en cuenta que, a partir de una primera partición con estas características, un algoritmo divisivo nunca llegará a la partición óptima dada por C_1, C_2 y C_3. Sin embargo, esta partición óptima de los tres grupos se puede obtener después de una primera iteración "regular" o no óptima (por ejemplo, $\{1, 2, 3, 4\}$ y $\{5, 6, 7, 8, 9, 10, 11, 12\}$).

Puesto que un algoritmo jerárquico divisivo muestra la evolución de las divisiones de los grupos en una red, no se puede tener en cuenta sólo una etapa, aunque es de esperar que las primeras iteraciones sean más relevantes que las últimas.

3.5.1. Red: "The Karate Club"

Uno de los ejemplos más conocidos de la literatura en las redes sociales o problemas de detección de comunidades es la red del Club de Karate ("The Karate Club network") definida por [118].

En esta red, los nodos representan los miembros de un club de karate y las aristas representan las amistades entre ellos. La red se compone de 34 miembros que se incorporan a un club de karate como nodos y 78 aristas que representan la amistad entre los miembros del club, tal y como se ha observado en un período de dos años. Debido a un desacuerdo entre el administrador y el instructor, el club se dividió en dos clubes más pequeños. La pregunta que nos preocupa es que si podemos descubrir el posible comportamiento de la red, detectar las dos comunidades o varios grupos, y en particular identificar a que comunidad pertenece cada nodo. Por lo general, la red del Club de Karate se estudia desde un punto de vista no valorado, sólo teniendo en cuenta el grafo asociado y las relaciones entre los miembros del Club de Karate. En la Figura 3.10 mostramos el rendimiento de los algoritmos *D&L, D&LF, GN, CNM, Radicchi, Walktrap y N2012* para el problema "The Karate Club network". Cada nivel del dendograma asociado por estos algoritmos corresponde a una partición de la red. Para cada nivel de partición $\mathcal{P}^t$ (que corresponden a un número fijo de comunidades) hemos medido su modularidad. En la red del Club de Karate, podemos observar que la mayoría de los algoritmos alcanzan las condiciones óptimas de partición, en términos de modularidad, en los primeros pasos (3, 4 o 5 comunidades).

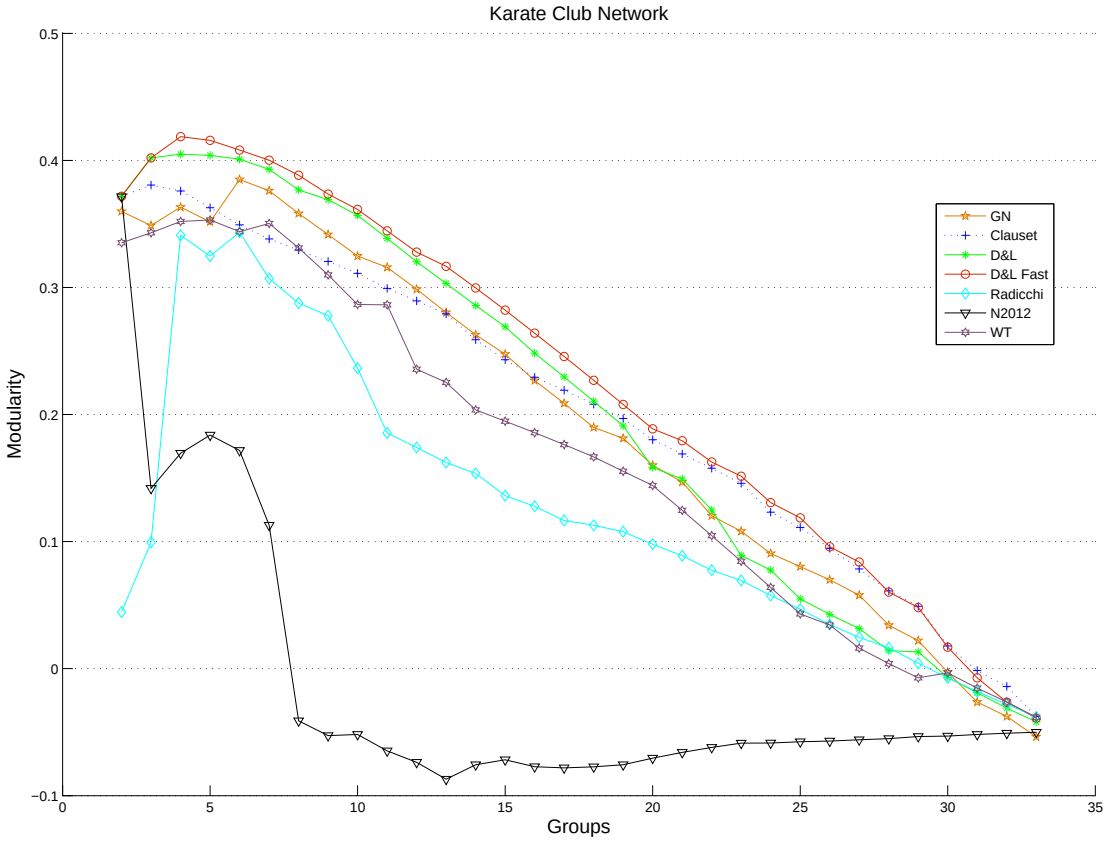


Figura 3.10: Red "The Karate Club": Comparación de resultados en términos de modularidad y número de grupos.

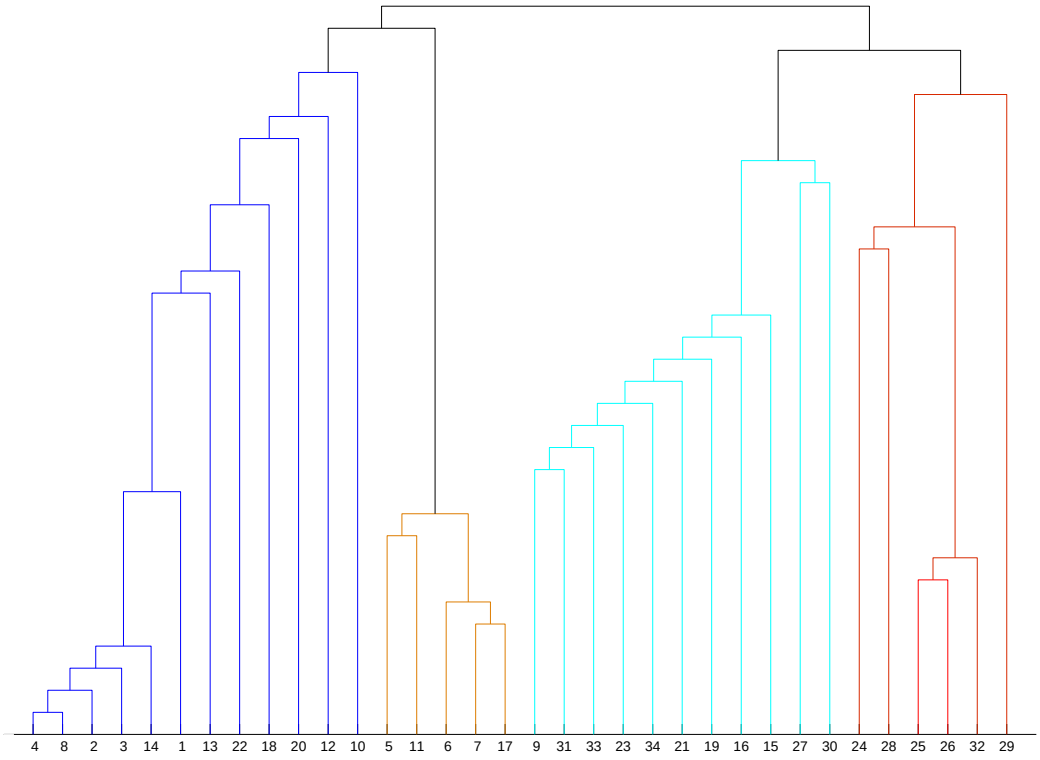


Figura 3.11: Red "The Karate Club": Dendograma.

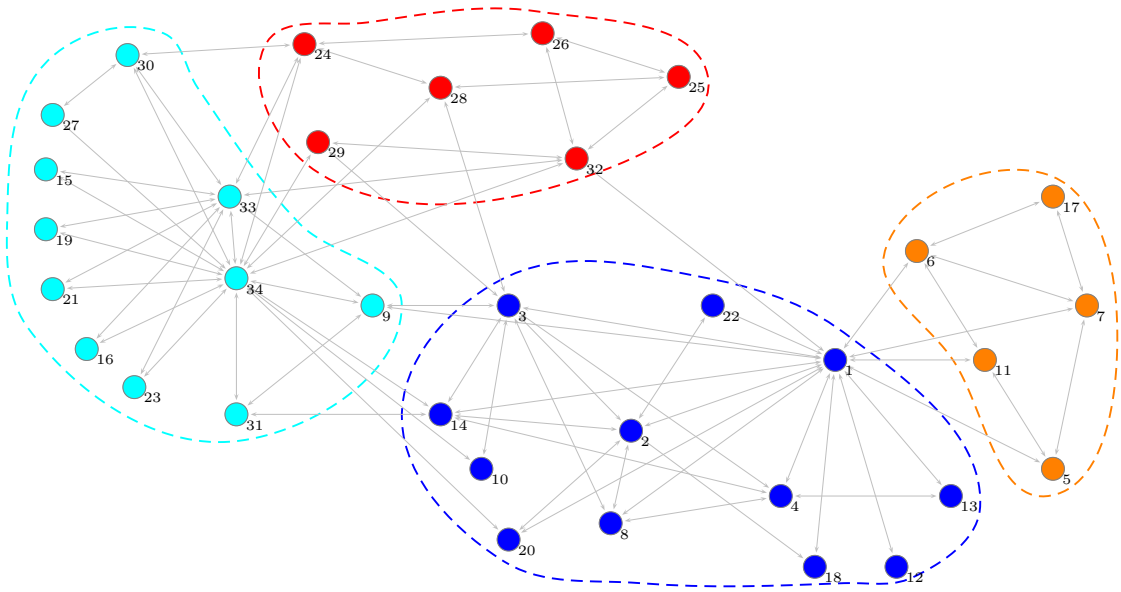


Figura 3.12: Red "The Karate Club": Cortes de máxima modularidad ($Q = 0{,}4050$) dada por el algoritmo Divide-and-Link.

Nos gustaría subrayar una vez más que no hay una manera única de comparar dos dendogramas, aunque podemos comparar las particiones en cada iteración, como se hizo

anteriormente con la modularidad. Podría pensarse en diferentes enfoques para comparar dos dendogramas, o dos vectores de modularidad (el máximo, la media, criterios lexicográficos, el promedio hasta el máximo, entre muchos otros. Teniendo en cuenta los criterios anteriores, se aprecia que los algoritmos *D&L y D&LF* presentan claramente los mejores resultados.

3.5.2. Red: "Les Miserables"

Otra red clásica utilizada para probar algoritmos de segmentación es la relacionada con la obra *Les Miserables*, en la que los nodos son los actores y suele estudiarse teniendo en cuenta el grafo asociado a las relaciones entre los actores. Los valores de estas relaciones pueden encontrarse en `http://www-personal.umich.edu/~mejn/netdata/`, y pueden entenderse como la afinidad o la fortaleza de tales relaciones.

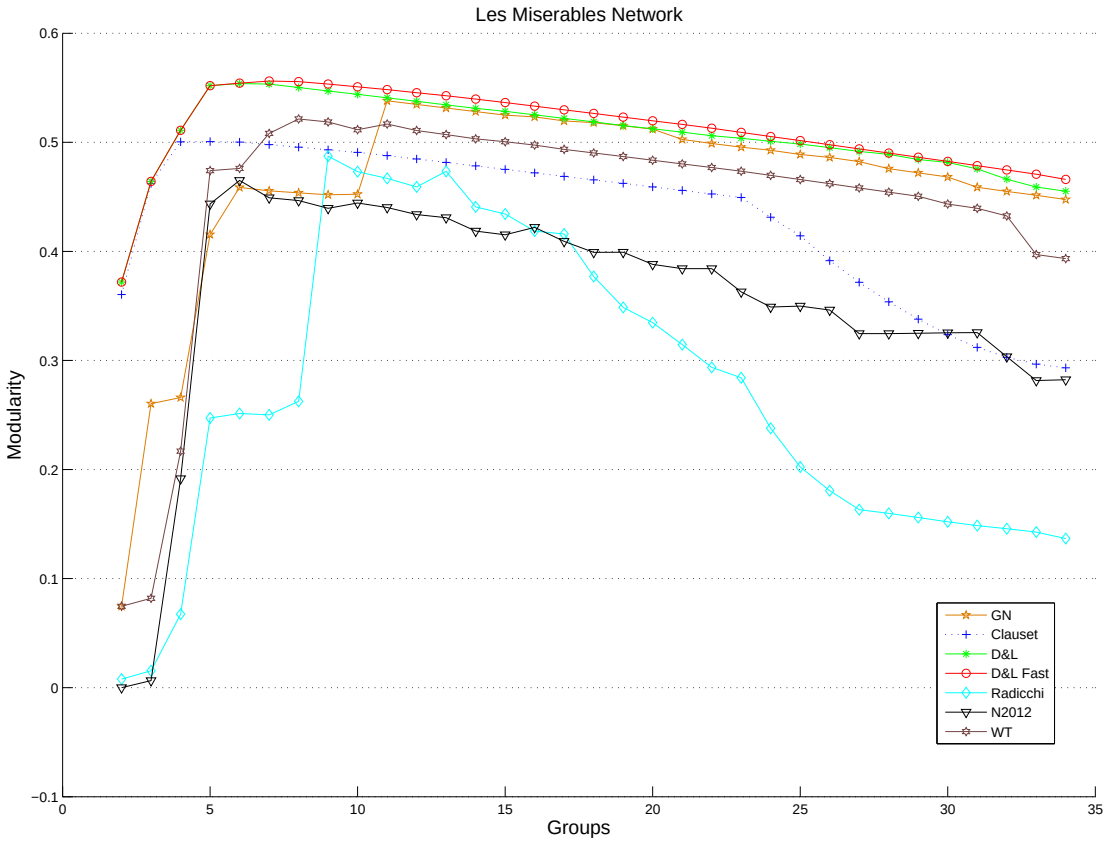


Figura 3.13: Red "Les Miserables": Modularidad vs grupos.

En la Figura 3.13 se muestra el rendimiento de los algoritmos *D&L, D&LF, GN, CNM, Radicchi, Walktrap* y *N2012* Podemos observar en esta red que la partición óptima (en términos de modularidad) se alcanza con menos de 11 comunidades. Observemos también que, al igual que en el ejemplo anterior, podemos concluir que los resultados dados por los algoritmos *D&L y D&LF* son, en general, de los mejores.

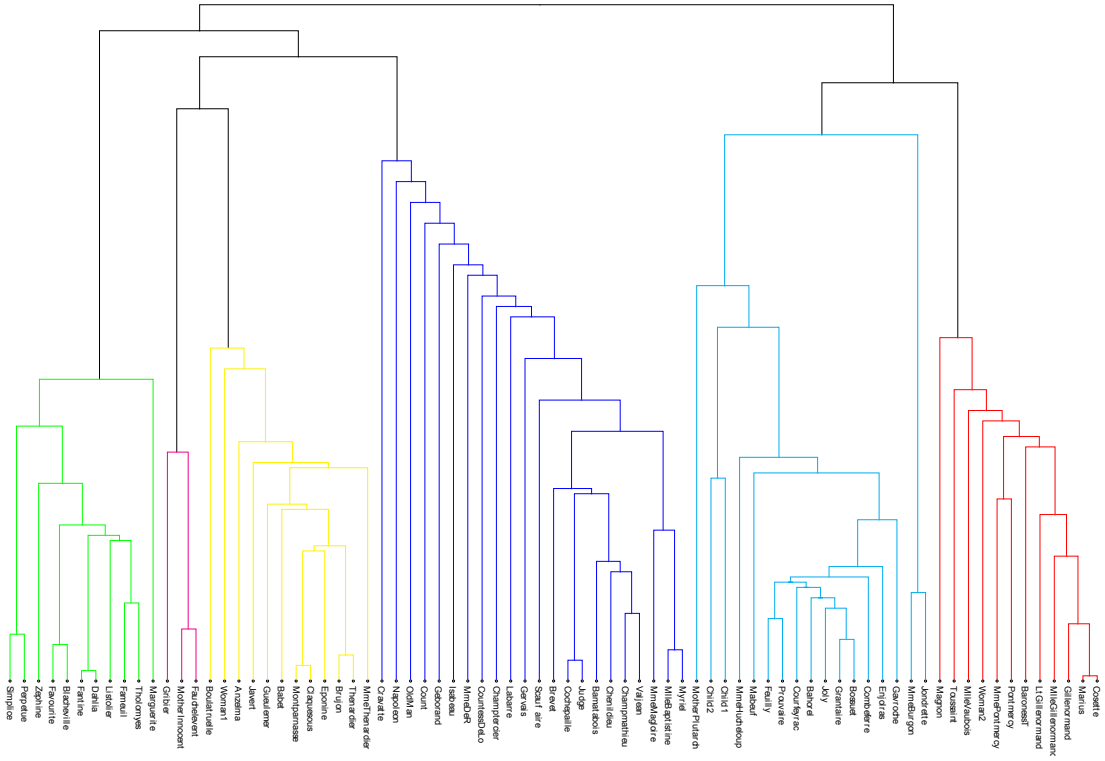


Figura 3.14: Red "Les Miserables": Dendograma.

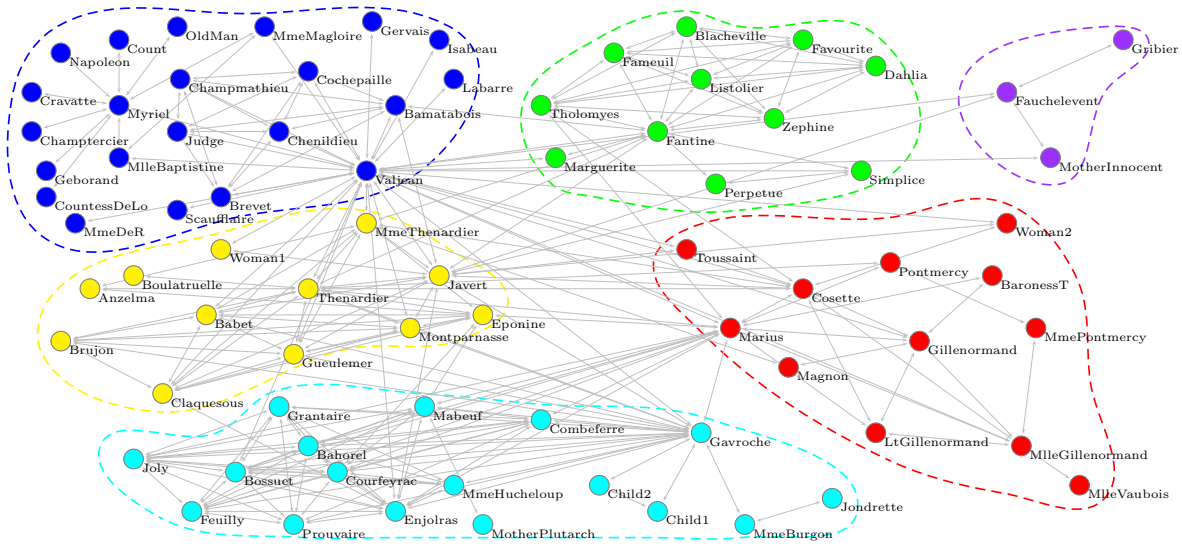


Figura 3.15: Red "Les Miserables": Corte máxima modularidad (Q=0.5539) obtenido por el algoritmo Divide-and-Link.

3.5.3. Red: "The authors"

"The centrality authors' network" o simplemente, la red "The authors" ha sido tomada desde el sitio web de Pajek `http://vlado. fmf.uni-lj.si/ pub/networks/pajek/`. La red analizada aquí comprende los autores de documentos que tratan sobre la centra-

lidad de la red y sus referencias cruzadas a partir de 1940 y hasta 1979. Las relaciones de la red representan las citas entre ellos. La red contiene 11 escritos que están aislados.

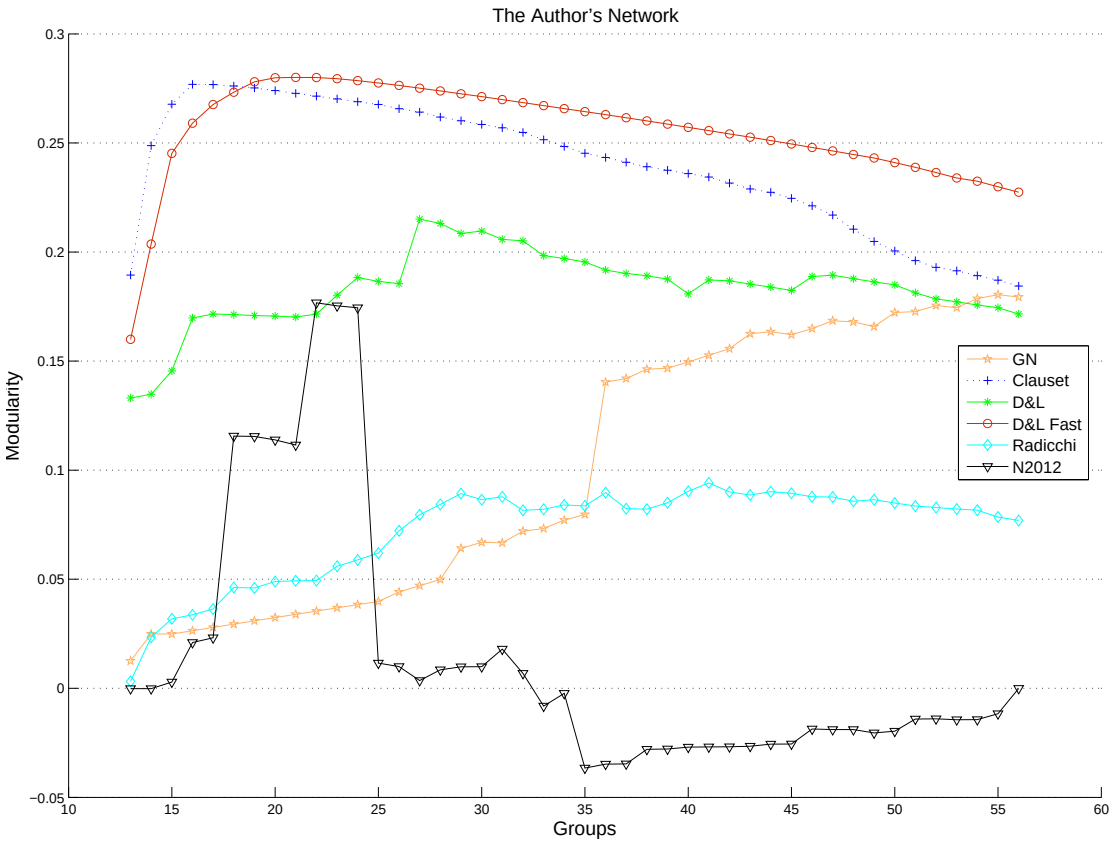


Figura 3.16: Red "The Authors": comparación de resultados en términos de modularidad y número de grupos.

En la Figura 3.16 se muestra el rendimiento, sobre la red "The authors", de los algoritmos *D&L, D&LF, GN, CNM, Radicchi* y *N2012* (no se considera el *algoritmo-Waltrap* ya que no trabaja con redes no conexas). Cada nivel del dendograma dado por estos algoritmos corresponde a una partición de la red, a partir de la partición trivial dada por las 12 componentes conexas de la red original. En cada nivel se ha medido la modularidad para la partición $\mathcal{P}^t$. En este caso podemos observar grandes diferencias entre los resultados. Pero nótese que esta red no presenta una estructura clara en términos de comunidades (una partición suele considerarse *buena* si su modularidad es mayor de $0,3$). A pesar de esto, se puede decir que el *algoritmo-D&LF* muestra un mejor rendimiento que los algoritmos *D&L, GN, Radicchi* y *N2012*, también puede verse que el *algoritmo-D&L* produce mejores resultados que los algoritmos *GN, Radicchi* y *N2012*. El *algoritmo-CNM* parece ser mejor que el *algoritmo-D&L*, sin embargo, no puede decirse lo mismo al compararlo con el *algoritmo-D&LF*. El *algoritmo-CNM* obtiene mejores particiones para las primeras cinco iteraciones; después de ésto, el *algoritmo-D&LF* produce mejores resultados. Si se utiliza para este ejemplo un criterio lexicográfico para comparar los algoritmos *CNM* vs *D&LF*, puede decirse que el *algoritmo-CNM* es mejor. Por el contrario, si el criterio de comparación fuera el máximo de las iteraciones, o el promedio en todas las iteraciones; entonces se consideraría al *algortimo-D&LF* como el mejor. Por lo tanto, no podemos concluir que alguno de ellos sea dominante.

La Figuras 3.16 y 3.17 muestran el desempeño de nuestro algoritmo de Divide-and-Link comparando los resultados con el algoritmo de división clásica desarrollado en [43].

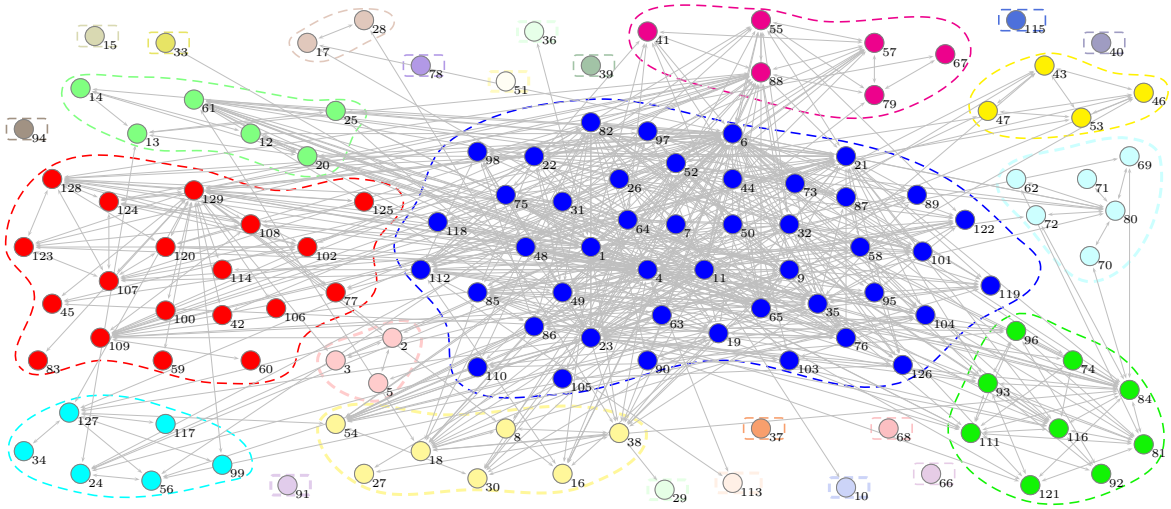


Figura 3.17: Red "The Authors": Corte de máxima modularidad (Q=0.2151) dado por el algoritmo Divide-and-Link.

También se muestra el corte de máxima modularidad dado por nuestro algoritmo y los cálculos correspondientes de la modularidad en la Tabla 3.6. Una vez más, podemos observar que los resultados obtenidos por el algoritmo de D&LF son mejores que el *algoritmo-GN* clásico, y requieren menos esfuerzos computacionales.

3.5.4. Red: "The dolphins"

Finalmente, la última red que vamos a analizar aquí es comúnmente llamado *The dolphin network* [70], también ampliamente analizada en la literatura detección de comunidades. Esta red fue construida a partir de observaciones de una comunidad de 62 delfines nariz de botella, y comprende 159 vínculos. Los nodos de la red representan los delfines y los lazos entre los nodos representan las asociaciones entre pares de delfines que ocurren con más frecuencia de lo esperado por el azar. Uno de los objetivos de esta red fue investigar el papel de los diferentes individuos para mantener la cohesión de las comunidades y de toda la red.

En la Figura 3.18 se muestra el rendimiento, en la red "The Dolphins", de los algoritmos *D&L, D&LF, GN, CNM, Radicchi, Walktrap* y *N2012*. En la primera iteración se pueden observar valores de modularidad muy parecidos. Sin embargo, en las iteraciones segunda y tercera, los mejores resultados corresponden a *D&L, D&LF y CNM* (note que en los algoritmos *GN, Radicchi, Walktrap* y *N2012* la modularidad decrece significativamente). En las posteriores iteraciones, el *algoritmo-D&LF* presenta muy buenos resultados, seguido por los algoritmos *GN, D&L* y *Walktrap*. Los algoritmos *CNM, Radicchi* y *N2012* no obtienen buenos resultados.

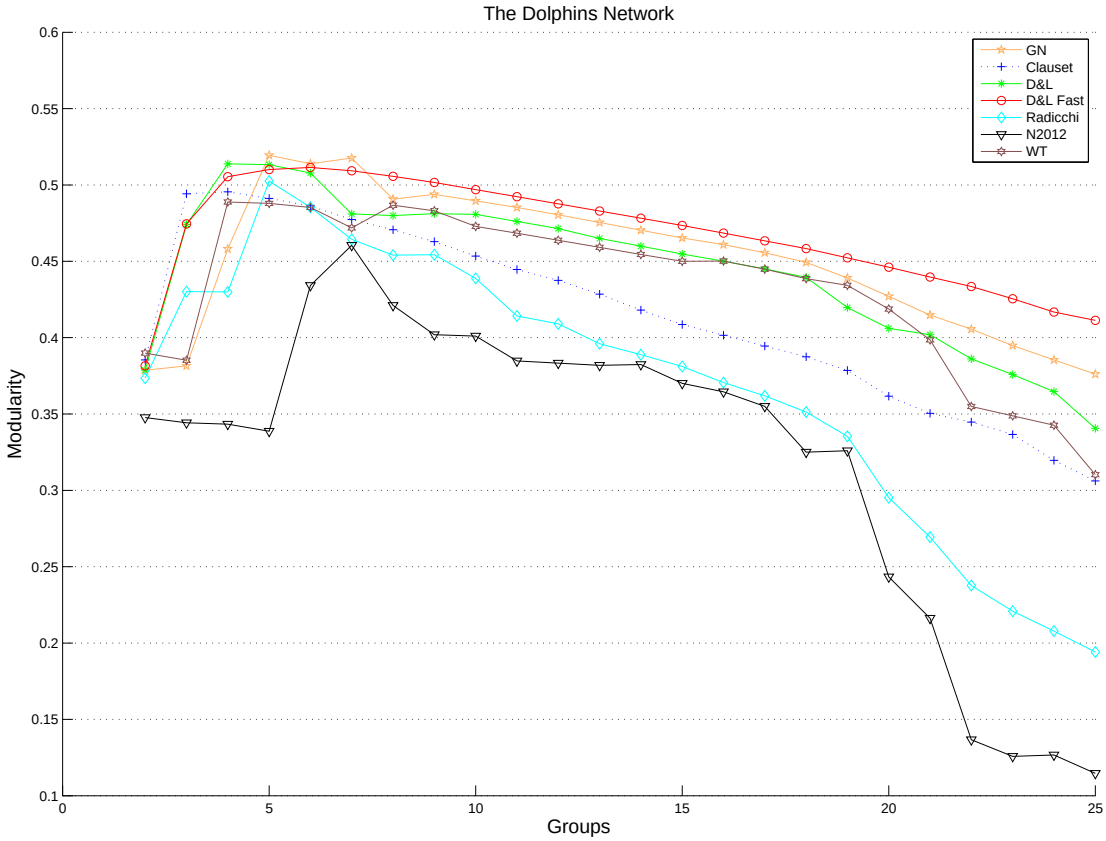


Figura 3.18: Red "The Dolphins": Comparación de resultados en términos de modularidad y número de grupos.

La Figura 3.18 y la Figura 3.19 muestran el rendimiento del algoritmo Divide-and-Link respecto a los algoritmos de división clásicos desarrollados en [43].

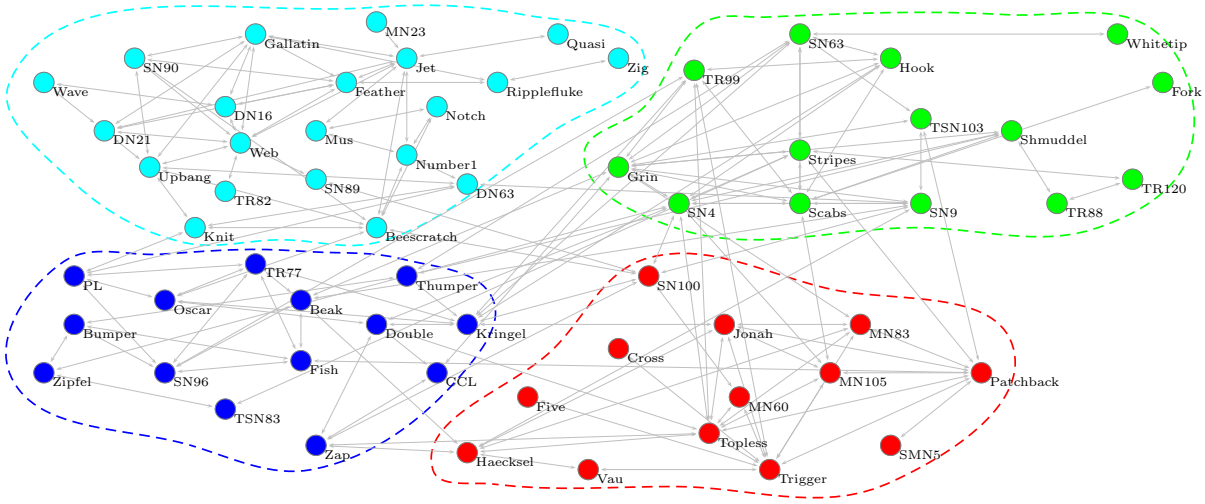


Figura 3.19: Red "The Dolphins": Corte de máxima modularidad (Q=0.5138) dado por el algoritmo Divide-and-Link.

3.6. Resultados con Redes Aleatorias

Con el fin de mostrar la robustez del *algoritmo D&L* y su versión rápida *D&LF*, hemos generado grafos aleatorios con 128 vértices y 4 comunidades $C_1, \ldots, C_4$, cada una con un tamaño $|C_i| = 32$, todo esto siguiendo el punto de vista de Girvan-Newman definido en [43]. El grado esperado para cualquier vértice es 16 (i.e. $< k >= 16$), pero el 'grado-out', definido como el número esperado de vecinos con vértice perteneciente a una comunidad diferente, y denotado como z_{out}, va desde 1 hasta 10. Por lo tanto, los valores más altos de grado-out corresponden a los grafos con estructuras de grupos más débiles. El experimento está diseñado para medir la sensibilidad de un algoritmo respecto a la cohesión de las comunidades. Por z_{in} denotamos el número esperado de vecinos de un vértice que pertenece a su comunidad. Para decidir si se debe incluir una arista, el grafo aleatorio es generado de acuerdo a la siguiente distribución de probabilidad para las aristas:

Dados dos nodos i y j, entonces,

$$P(i,j) \;=\; \begin{cases} \alpha & \text{si} \quad i \text{ y } j \text{ pertenecen a} \quad C_k \\ \beta & \text{en otra parte.} \end{cases} \tag{3.9}$$

El parámetro de mezcla, μ, se define como:

$$\mu \;=\; \frac{z_{\text{out}}}{z_{\text{in}} + z_{\text{out}}}. \tag{3.10}$$

El parámetro μ dado en (3.10) representa el porcentaje de enlaces por fuera de la comunidad. Para cualquier nodo i, se tiene que $< k_i >= 16 = z_{\text{in}} + z_{\text{out}} = 31\alpha + 96\beta$ de manera que, a partir del valor de μ o z_{out}, un grafo pueda ser generado dentro del marco de *GN*. En general, (ver [68] para más detalles), la mayoría de los algoritmos, por ejemplo *GN, Radicchi, Walktrap, Blondel, DM* o *CNM*, fallan para valores de μ mayores que 0,4, pero son capaces de encontrar a las comunidades $C_1, \ldots, C_4$ para valores inferiores de $\mu = 0{,}4$.

Con el fin de medir la similitud entre una partición determinada y la partición de referencia $\mathcal{P} = \{C_1, \ldots, C_4\}$ con $C_1 = \{1, \ldots, 32\}$, $C_2 = \{33, \ldots, 64\}$, $C_3 = \{65, \ldots, 96\}$ and $C_4 = \{97, \ldots, 128\}$, un algoritmo que suele usarse es la "Información Mutua Normalizada", NMI (Normalized Mutual Information). Este algoritmo representa, en cierta forma, el porcentaje de nodos clasificados correctamente. La medida de NMI para un conjunto $\Omega = \{\omega_1, \ldots, \omega_r\}$ de grupos, y un conjunto $\Lambda = \{\lambda_1, \ldots, \lambda_j\}$ de clases viene dada por:

$$\text{NMI} \;=\; \frac{I(\Omega;\Lambda)}{[H(\Omega) + H(\Lambda)]/2} \tag{3.11}$$

donde $I(\Omega;\Lambda)$ es la información mutua y H es la entropía, calculadas así:

$$I(\Omega;\Lambda) = \sum_k \sum_j P(\omega_k \cap \lambda_j) \log \frac{P(\omega_k \cap \lambda_j)}{P(\omega_k) P(\lambda_j)}$$

y

$$H(\Omega) = -\sum_k P(\omega_k) \log P(\omega_k)$$

donde $P(\omega_k)$, $P(\lambda_j)$x y $P(\omega_k \cap \lambda_j)$ son las probabilidades de pertenecer al grupo ω_k, la clase λ_j, y a la intersección de ω_k y λ_j, respectivamente.

En la Tabla 3.3, cada punto corresponde a un promedio de más de 100 realizaciones de referencia. Hemos incluido resultados de otros algoritmos como *Blondel, C-finder, Radicchi, CNM*, y el algoritmo *Doneti-Muñoz* (ver [26]), para mostrar la eficacia de nuestro algoritmo. Algunos de estos resultados han sido tomados de [68].

μ	z_{out}	z_{in}	**D&L**	**D&LF**	**CNM**	**GN**	**Blondel**	**Radicchi**	**N2012**	***C*-finder**
0.1	1.6	14.4	1	1	1	1	1	0.87	1	0.78
0.2	3.2	12.8	1	1	0.9799	1	1	0.9	1	0.5
0.3	4.8	11.2	1	0.9596	0.8031	0.9	0.9	0.78	0.9469	0.3
0.35	5.6	10.4	1	0.8777	0.7667	0.85	0.9	0.5	0.8623	0.27
0.40	6.4	9.6	0.8358	0.7095	0.5929	0.42	0.78	0.1	0.7413	0.2
0.45	7.2	8.8	0.6333	0.4792	0.3885	0.23	0.56	0	0.5547	0.16
0.5	8.0	8	0.4028	0.2681	0.3666	0.1	0.27	0	0.3055	0.1
0.55	8.8	7.2	0.234	0.1462	0.1059	0.01	0.1	0	0.1366	0

Tabla 3.3: Información Mutua Normalizada (NMI) entre la partición de referencia con 4 grupos y las particiones dadas por el *algoritmo-D&L*, su versión *D&LF* y otros algoritmos ampliamente conocidos para grafos aleatorios de 128 nodos definidos según [43] en función del parámetro μ.

Como lo muestra la Tabla 3.3, el *algoritmo-D&L* muestra los mejores resultados. Ambos algoritmos, *D&L* y *D&LF* obtienen muy buenos resultados y valores de NMI similares para $\mu < 0{,}3$, pero para $\mu \geq 0,35$ las diferencias entre ellos son significativas. Analizando los resultados de la Tabla 3.3, podemos decir que el rendimiento de los dos algoritmos, *D&L* y *D&LF*, que hemos propuesto, se comportan, en general, mejor que los otros algoritmos clásicos que se han considerado para la comparación.

3.7. Valores de Modularidad en Redes Sociales

Tabla 3.4: The Karate Club Network: Modularidad

	GN	Clauset	D&L	D&L Fast	Radicchi	N2012	WT
2	0,35996	0,37179	0,37180	0,37180	0,04463	**0,37147**	0,33522
3	0,34878	**0,38067**	0,40204	0,40204	0,09952	0,14193	0,34311
4	0,36325	0,37599	**0,40500**	**0,41880**	0,34114	0,16954	0,35199
5	0,35174	0,36284	0,40401	0,41585	0,32479	0,18384	**0,35322**
6	**0,38503**	0,34936	0,40105	0,40820	**0,34352**	0,17176	0,34402
7	0,37623	0,33826	0,39308	0,40023	0,30703	0,11284	0,35043
8	0,35832	0,32955	0,37681	0,38840	0,28780	-0,04109	0,33120
9	0,34172	0,32051	0,36917	0,37360	0,27794	-0,05276	0,30999
10	0,32479	0,31114	0,35684	0,36144	0,23652	-0,05178	0,28665
11	0,31591	0,29931	0,33876	0,34467	0,18549	-0,06476	0,28641
12	0,29865	0,28945	0,32035	0,32791	0,17415	-0,07372	0,23570
13	0,28041	0,27926	0,30325	0,31673	0,16231	-0,08703	0,22535
14	0,26282	0,25896	0,28583	0,29964	0,15360	-0,07569	0,20365
15	0,24754	0,24318	0,26907	0,28222	0,13618	-0,07175	0,19477
16	0,22682	0,22937	0,24836	0,26414	0,12779	-0,07717	0,18573
17	0,20891	0,21918	0,22962	0,24573	0,11662	-0,07816	0,17636
18	0,18984	0,20817	0,21055	0,22699	0,11284	-0,07733	0,16667
19	0,18130	0,19699	0,19116	0,20792	0,10782	-0,07569	0,15533
20	0,16009	0,18023	0,15837	0,18877	0,09796	-0,07051	0,14415
21	0,14694	0,16905	0,14924	0,17941	0,08884	-0,06591	0,12459
22	0,12032	0,15771	0,12492	0,16264	0,07750	-0,06197	0,10470
23	0,10815	0,14587	0,08909	0,15146	0,06936	-0,05868	0,08448
24	0,09065	0,12319	0,07758	0,13075	0,05753	-0,05851	0,06394
25	0,08029	0,11111	0,05490	0,11867	0,04684	-0,05753	0,04306
26	0,06994	0,09467	0,04274	0,09599	0,03468	-0,05703	0,03435
27	0,05786	0,07848	0,03156	0,08391	0,02465	-0,05605	0,01611
28	0,03419	0,06123	0,01414	0,06024	0,01644	-0,05506	0,00394
29	0,02202	0,04906	0,01315	0,04808	0,00427	-0,05342	-0,00723
30	-0,00288	0,01775	-0,00690	0,01685	-0,00723	-0,05309	-0,00329
31	-0,02630	-0,00148	-0,01898	-0,00723	-0,01808	-0,05178	-0,01537
32	-0,03764	-0,01397	-0,03107	-0,02646	-0,02794	-0,05079	-0,02622
33	-0,05375	-0,03764	-0,04191	-0,03895	-0,03797	-0,05013	-0,03830

Tabla 3.5: Les Miserables Network: Modularidad

	GN	**Clauset**	**D&L**	**D&L Fast**	**Radicchi**	**N2012**	**WT**
2	0,07464	0,36046	0,37185	0,37185	0,00780	-0,00001	0,07464
3	0,26041	0,46228	0,46423	0,46423	0,01546	0,00646	0,08191
4	0,26605	0,50046	0,51104	0,51104	0,06742	0,19159	0,21683
5	0,41547	**0,50060**	0,55199	0,55199	0,24728	0,44356	0,47408
6	0,45872	0,50017	**0,55386**	0,55426	0,25140	**0,46501**	0,47605
7	0,45545	0,49786	0,55343	**0,55613**	0,25006	0,44917	0,50817
8	0,45366	0,49552	0,55027	0,55570	0,26256	0,44656	**0,52141**
9	0,45187	0,49315	0,54711	0,55339	**0,48727**	0,43945	0,51868
10	0,45242	0,49077	0,54394	0,55087	0,47310	0,44441	0,51159
11	**0,53807**	0,48786	0,54077	0,54828	0,46692	0,44027	0,51680
12	0,53478	0,48470	0,53758	0,54551	0,45913	0,43381	0,51084
13	0,53149	0,48153	0,53439	0,54269	0,47339	0,43090	0,50711
14	0,52819	0,47835	0,53119	0,53962	0,44075	0,41865	0,50322
15	0,52488	0,47517	0,52841	0,53649	0,43425	0,41519	0,50057
16	0,52336	0,47198	0,52520	0,53316	0,41856	0,42197	0,49748
17	0,51959	0,46878	0,52199	0,52981	0,41599	0,40922	0,49356
18	0,51804	0,46557	0,51876	0,52647	0,37693	0,39912	0,49030
19	0,51516	0,46235	0,51542	0,52312	0,34869	0,39921	0,48695
20	0,51213	0,45913	0,51244	0,51976	0,33468	0,38805	0,48359
21	0,50271	0,45589	0,50932	0,51640	0,31457	0,38418	0,48023
22	0,49893	0,45266	0,50609	0,51289	0,29364	0,38421	0,47686
23	0,49558	0,44941	0,50377	0,50912	0,28412	0,36287	0,47348
24	0,49267	0,43129	0,50095	0,50535	0,23783	0,34894	0,46967
25	0,48889	0,41424	0,49838	0,50157	0,20234	0,34980	0,46585
26	0,48610	0,39144	0,49514	0,49778	0,18060	0,34625	0,46202
27	0,48231	0,37164	0,49165	0,49398	0,16318	0,32462	0,45818
28	0,47583	0,35373	0,48900	0,49017	0,15984	0,32464	0,45434
29	0,47203	0,33790	0,48419	0,48636	0,15603	0,32497	0,45049
30	0,46823	0,32366	0,48156	0,48250	0,15211	0,32534	0,44340
31	0,45876	0,31204	0,47562	0,47860	0,14855	0,32559	0,43954
32	0,45495	0,30292	0,46616	0,47469	0,14572	0,30342	0,43248
33	0,45145	0,29663	0,45903	0,47077	0,14262	0,28155	0,39708
34	0,44759	0,29324	0,45514	0,46596	0,13666	0,28236	0,39334

Tabla 3.6: Modularidad para la red: Autores

	GN	**Clauset**	**D&L**	**D&L Fast**	**Radicchi**	**N2012**	**WT***
13	0,01254	0,18946	0,13310	0,15998	0,00320	-0,00016	
14	0,02493	0,24879	0,13480	0,20361	0,02342	-0,00018	
15	0,02489	0,26778	0,14550	0,24522	0,03187	0,00283	
16	0,02641	**0,27685**	0,16980	0,25907	0,03371	0,02097	
17	0,02792	0,27676	0,17160	0,26758	0,03634	0,02314	
18	0,02943	0,27612	0,17130	0,27327	0,04621	0,11563	
19	0,03093	0,27521	0,17090	0,27804	0,04597	0,11546	
20	0,03243	0,27403	0,17060	0,27990	0,04894	0,11387	
21	0,03393	0,27276	0,17020	**0,28003**	0,04930	0,11154	
22	0,03541	0,27149	0,17160	0,28002	0,04941	**0,17671**	
23	0,03690	0,27021	0,18020	0,27947	0,05595	0,17533	
24	0,03837	0,26894	0,18830	0,27855	0,05887	0,17444	
25	0,03978	0,26766	0,18650	0,27753	0,06199	0,01155	
26	0,04415	0,26570	0,18550	0,27640	0,07217	0,00994	
27	0,04705	0,26417	**0,21510**	0,27512	0,07953	0,00353	
28	0,04984	0,26188	0,21310	0,27384	0,08432	0,00847	
29	0,06415	0,26019	0,20850	0,27255	0,08931	0,00987	
30	0,06691	0,25848	0,20960	0,27125	0,08647	0,00991	
31	0,06669	0,25692	0,20580	0,26988	0,08778	0,01794	
32	0,07199	0,25483	0,20510	0,26850	0,08161	0,00681	
33	0,07321	0,25149	0,19840	0,26713	0,08207	-0,00817	
34	0,07707	0,24842	0,19700	0,26575	0,08402	-0,00232	
35	0,07971	0,24527	0,19550	0,26437	0,08364	-0,03658	
36	0,14034	0,24333	0,19170	0,26297	0,08972	-0,03474	
37	0,14200	0,24113	0,19020	0,26155	0,08234	-0,03464	
38	0,14626	0,23910	0,18910	0,26011	0,08213	-0,02799	
39	0,14676	0,23752	0,18760	0,25865	0,08504	-0,02783	
40	0,14961	0,23595	0,18080	0,25717	0,09030	-0,02696	
41	0,15267	0,23438	0,18720	0,25568	**0,09416**	-0,02686	
42	0,15568	0,23160	0,18680	0,25416	0,09002	-0,02680	
43	0,16251	0,22894	0,18530	0,25264	0,08861	-0,02656	
44	0,16345	0,22736	0,18390	0,25109	0,09013	-0,02559	
45	0,16212	0,22463	0,18240	0,24953	0,08938	-0,02552	
46	0,16489	0,22116	0,18880	0,24796	0,08779	-0,01868	
47	0,16852	0,21693	0,18940	0,24635	0,08770	-0,01894	
48	0,16799	0,21045	0,18780	0,24473	0,08573	-0,01891	
49	0,16581	0,20484	0,18630	0,24311	0,08646	-0,02047	

Modularidad para la red: Autores (continuación)

	GN	**Clauset**	**D&L**	**D&L Fast**	**Radicchi**	**N2012**	**WT***
50	0,17228	0,20053	0,18500	0,24097	0,08492	-0,01964	
51	0,17269	0,19606	0,18130	0,23881	0,08356	-0,01410	
52	0,17542	0,19298	0,17850	0,23642	0,08288	-0,01397	
53	0,17452	0,19140	0,17720	0,23397	0,08220	-0,01440	
54	0,17867	0,18915	0,17570	0,23245	0,08163	-0,01437	
55	**0,18043**	0,18709	0,17450	0,22994	0,07853	-0,01165	
56	0,17936	0,18440	0,17160	0,22742	0,07691	-0,00006	

Tabla 3.7: Modularidad para la red: Delfines

	GN	**Clauset**	**D&L**	**D&L Fast**	**Radicchi**	**N2012**	**WT**
2	0,37870	0,38543	0,37870	0,38155	0,37348	0,34767	0,38986
3	0,38149	0,49419	0,47410	0,47455	0,43015	0,34423	0,38519
4	0,45807	**0,49549**	**0,51380**	0,50540	0,42995	0,34332	**0,48885**
5	**0,51938**	0,49120	0,51330	0,51009	**0,50257**	0,33869	0,48796
6	0,51392	0,48629	0,50780	**0,51147**	0,48538	0,43418	0,48527
7	0,51756	0,47731	0,48100	0,50930	0,46438	**0,46048**	0,47186
8	0,49072	0,47067	0,48000	0,50568	0,45398	0,42109	0,48669
9	0,49381	0,46288	0,48120	0,50160	0,45433	0,40194	0,48307
10	0,48956	0,45340	0,48070	0,49695	0,43877	0,40105	0,47288
11	0,48529	0,44462	0,47610	0,49229	0,41420	0,38476	0,46842
12	0,48046	0,43752	0,47140	0,48762	0,40904	0,38333	0,46379
13	0,47542	0,42846	0,46500	0,48293	0,39599	0,38191	0,45914
14	0,47035	0,41802	0,45990	0,47823	0,38887	0,38240	0,45447
15	0,46527	0,40859	0,45480	0,47350	0,38108	0,37006	0,44998
16	0,46092	0,40159	0,45010	0,46846	0,37059	0,36454	0,45016
17	0,45558	0,39455	0,44500	0,46339	0,36189	0,35499	0,44486
18	0,44931	0,38748	0,43950	0,45831	0,35129	0,32501	0,43863
19	0,43907	0,37855	0,41970	0,45230	0,33539	0,32590	0,43424
20	0,42708	0,36169	0,40610	0,44613	0,29516	0,24331	0,41873
21	0,41478	0,35042	0,40190	0,43972	0,26935	0,21621	0,39856
22	0,40558	0,34473	0,38610	0,43351	0,23767	0,13670	0,35489
23	0,39482	0,33660	0,37580	0,42544	0,22082	0,12581	0,34880
24	0,38541	0,31963	0,36470	0,41674	0,20782	0,12670	0,34269
25	0,37615	0,30614	0,34060	0,41134	0,19410	0,11461	0,31029

Capítulo 4

Segmentación de Imágenes Digitales

Contenido

La segmentación de imágenes es una de las áreas de investigación más importantes de la visión por ordenador y sus aplicaciones, tales como el reconocimiento de patrones, la medicina, la robótica y la industria. Estas técnicas son bien conocidos por su utilidad y complejidad. La segmentación clásica de imágenes digitales es asociada frecuentemente a la búsqueda de 'objetos' cuando los bordes no son claros. Es natural que las imágenes no muestren bordes claros o formas estándares; lo que suele suceder, es encontrarse con una 'degradación' de una región a otra.

Algunos problemas de optimización y clasificación de imágenes pueden modelarse como un problema de grafos. En este capítulo mostraremos una aplicación del Algoritmo

Divide-and-Link, introducido en el Capítulo 2, para segmentar imágenes de manera jerarquizada. Este desarrollo está basado en la aplicación sucesiva de un coloreado binario, que produce a través de diferentes valores de umbrales, la segmentación de una imagen. Esta técnica realiza particiones de todos los pixeles, así cada pixel estará en una y sólo una 'clase de color'.

Un cambio de color sugerirá únicamente una región diferente. Cada una de estas regiones sugiere una clase (región 'homogénea'), sin embargo el color puede o no estar asociado a la clase. Una comparación detallada entre diferentes regiones podrá entonces ser desarrollada, realizando entonces una clasificación de bastante utilidad determinando posibles clases y sus zonas de transición.

Los algoritmos de segmentación son bien conocidos en el campo del procesamiento de imágenes. En este capítulo se propone un algoritmo polinomial y eficiente para la segmentación de imágenes que puede utilizar distancias nítidas o borrosas permitiendo particiones difusas. La principal diferencia entre los algoritmos de segmentación clásica está en los resultados entregados por el proceso de segmentación. Dado que las 'salidas' de los algoritmos clásicos de segmentación entregan regiones homogéneas en la imagen, el propósito de este procedimiento es entregar una información jerarquizada (de la misma forma como un dendograma lo hace en los métodos clásicos de clustering) de cómo los grupos son formados en la imagen, desde una situación inicial en la cual todos los pixeles pertenecen al mismo grupo hasta la situación final en la cual la imagen real es dividida en las unidades mínimas de información.

4.1. Introducción

La segmentación de imágenes puede entenderse como un caso particular de un proceso de clustering en el cual la estructura que representa la relación entre los pixeles no puede olvidarse. En el campo de la segmentación de imágenes es obvio que la estructura contenida en la imagen debe ser tenida en cuenta. Al respecto, la información contenida en cada pixel puede describirse en dos:

1. **Atributos:** describen características intrínsecas de cada pixel.
2. **Relación:** representan las relaciones (en este caso, en términos de vecindad) entre los pixeles.

Para el segundo caso, las relaciones pueden expresarse como un grafo. Estos problemas de "clustering" pueden desarrollarse como un problema de partición de grafos. Entre algunas aplicaciones relacionadas con lo anterior podemos enumerar:

1. Definición de zonas electorales de un país.
2. Análisis de redes sociales.
3. Establecimiento de regiones de trabajo (por ejemplo, para vendedores).

En general, para obtener particiones (o segmentaciones) en un grafo, en un periodo razonable de tiempo, usualmente es necesario imponer muchas restricciones asociadas al grafo. En análisis de redes sociales, se pueden encontrar algunas referencias que tratan de manera conjunta un análisis cluster con 'atributos' y 'relaciones' de los datos

haciendo énfasis en la importancia de esta problemática (ver, por ejemplo, [31] y [106]). Desafortunadamente, estas metodologias presentan problemas similares a los problemas clásicos de segmentación desde el punto de vista computacional o de la información final dada por la segmentación.

En este capítulo se continua la investigación iniciada en [49], donde una imagen fue concebida como un grafo de pixeles que admite una representación en el plano; así, los procesos de segmentación posteriores tendrán en cuenta la teoría de grafos para realizar la segmentación en regiones homogéneas con una clasificación no supervisada por medio de un algoritmo de coloreado (ver [50]). En particular, el objetivo principal de este trabajo es ofrecer una metodología alternativa que supere las limitaciones de algoritmos previos como la complejidad computacional, siendo válida, en general, para grafos arbitrarios, con una topología más compleja. En esta dirección, los algoritmos pueden ser de gran utilidad en problemas de clasificación donde las unidades de información no son independientes y definen un grafo.

La principal contribución del desarrollo que presentaremos a continuación es construir una segmentación jerárquica de una imagen en la cual los pixeles que van a ser agrupados no son independientes; y las relaciones entre dichos pixeles son dadas por un grafo (que puede ser 'borroso' o 'nítido').

Posteriormente, desarrollaremos una técnica jerárquica y eficiente de segmentación, la cual pertenece a la clasificación no supervisada, para pixeles relacionados a través de un grafo. La salida de este algoritmo muestra la evolución jerárquica (particiones) de la segmentación dividiendo desde la primera iteración en la que todos los pixeles están en el mismo grupo hasta la última iteración en la que cada grupo tiene un solo elemento.

4.1.1. La Imagen digital

La imagen puede ser entendida como una estructura bidimensional de pixeles, cada uno estando conectado con sus 'vecinos' naturales en la imagen. Por supuesto, cada pixel será caracterizado por un número fijo de atributos medibles. Estos atributos pueden ser, por ejemplo; los valores de tres bandas del espectro visible (rojo, verde y azul), o una familia de espectros de bandas de intensidad, o cualquier otra familia de medidas físicas. Considere el conjunto de píxeles

$$V = \{\ (i,j)\ \} \qquad i = 1, 2, \ldots, r \quad \text{y} \quad j = 1, 2, \ldots, s \tag{4.1}$$

descritos por medio de sus coordenadas cartesianas (i, j) en una imagen de tamaño $r \times s$. De esta forma, la información acerca de una imagen puede resumirse como:

$$I = \left\{\ (x^1_{i,j}, x^2_{i,j}, \ldots, x^b_{i,j})\ \mid\ (i,j) \in V\ \right\}. \tag{4.2}$$

En este caso, cada pixel (i, j) está siendo caracterizado por b medidas numéricas. Así por ejemplo, para el caso de imágenes RGB estaremos trabajando con $b = 3$. Los algoritmos de segmentación pueden ser clasificados, principalmente, en las siguientes clases: "clustering" (ver [115, 24, 73, 104]), detección de bordes (ver [19, 102, 20]), regiones (ver [99, 96, 86]), histogramas (ver [71, 116]) y técnicas basadas en grafos.

Las técnicas de segmentación de imágenes "basadas en grafos", pueden categorizarse en las siguientes dos clases:

1. Métodos de segmentación de imágenes **supervisados**:

- Modelos Mínimo corte / Máximo flujo (ver [7, 8, 85, 113]),
- Modelos de caminos aleatorios (ver [54, 55, 57, 3]).

2. Métodos de segmentación de imágenes **no supervisados**:

- Mínimo árbol soporte (ver [36]),
- Corte normalizado (ver [96, 110, 111, 63])
- Partición isoperimétrica de grafos (ver [56]).

El algoritmo que presentaremos a continuación puede ser clasificado como una técnica no supervisada de segmentación de imágenes basada en grafos. El enfoque jerárquico que se propondrá, utiliza los elementos claves del algoritmo de segmentación propuesto en [49] y que se amplió en [48]. El enfoque ofrecido por el algoritmo Divide-and-Link introducido el Capítulo 2, será utilizado en el tratamiento de imágenes digitales con el propósito de mejorar la complejidad computacional de los algoritmos propuestos en [49] y [48]; además de permitirnos analizar imágenes de gran tamaño.

4.2. Divide-and-Link en Segmentación Jerárquica de Imágenes

Inicialmente, al considerar una imagen como un grafo, los nodos serán entonces los pixeles y las aristas sus relaciones, según algún criterio predefinido. Formalmente, sea

$$V = \{P_1, P_2, \ldots, P_n\} \tag{4.3}$$

el conjunto (finito) de los pixeles de una imagen. También, sea

$$E = \big\{\{P_a, P_b\} \;\mid\; P_a,\, P_b \in V\big\} \tag{4.4}$$

el conjunto de pares no-ordenados de pixeles "vecinos" que están relacionados: si dos pixeles P_a, $P_b \in V$ están relacionados, entonces existe una arista $e_{ab} = \{P_a, P_b\} \in E$; en caso contrario entonces $\{P_a, P_b\} \notin E$. Por lo tanto, usando (4.3) y (4.4) podemos entonces definir un grafo

$$G = (V, E) \tag{4.5}$$

para modelar las relaciones entre los pixeles de la imagen. El grafo G dado en (4.5) puede ser asumido conexo, pues en caso contrario, se puede trabajar con cada componente conexa de manera separada. Una topología ampliamente usada para imágenes es aquella donde un pixel es vinculado con cuatro vecinos (ver la Figura. 4.1(a)); otras topologías son mostradas en las Figuras 4.1 (b), (c) and (d). Nosotros, por simplicidad, hemos escogido una conectividad de cuatro vecinos de la Figura. 4.1(a), para explicar nuestro procedimiento.

Dada $e_{ab} = \{P_a, P_b\} \in E$, una arista que une a los pixeles P_a y P_b, Sea $d_{ab} \geq 0$ el grado de disimiliaridad entre ellos: valores altos de d_{ab} implican mayor disimilaridad entre los los pixeles P_a and P_b . Luego, la matriz

$$D = \{d_{ab} \;\mid\; e_{ab} \in E\} \tag{4.6}$$

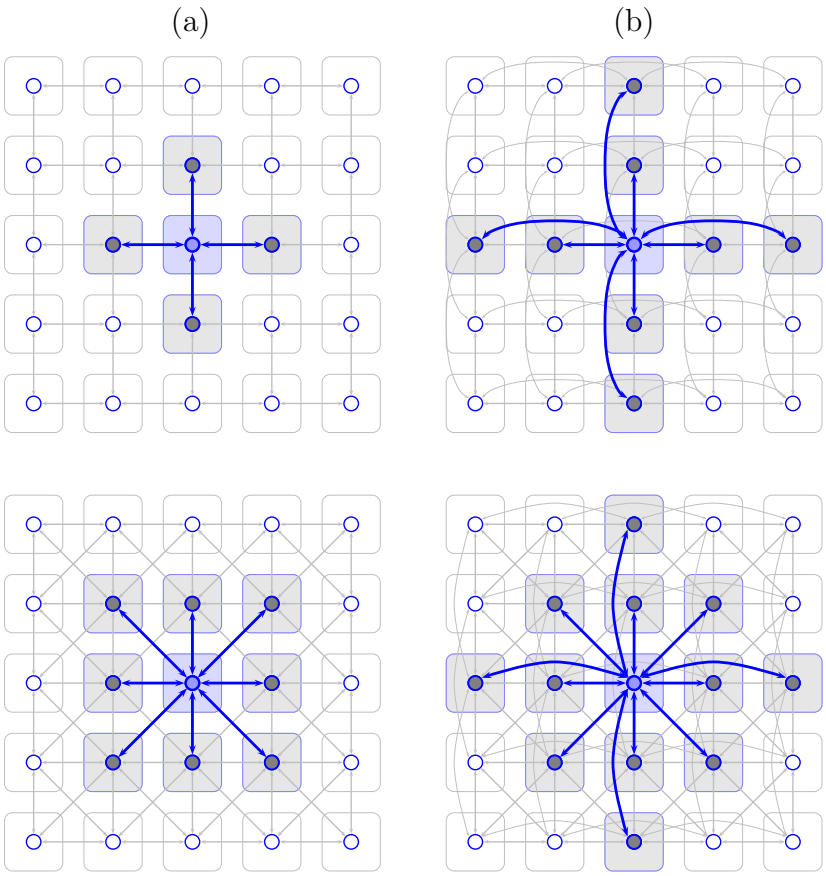


Figura 4.1: Redes. (a): 4 vecinos, (b-c): 8 vecinos, y (d) 12 vecinos.

reune todas estas disimilaridades Esta medida de disimilaridad suele definirse teniendo en cuenta el problema específico y el características de los elementos considerados. Su construcción y propiedades no hacen parte de los objetivos de este capítulo. Finalmente, usando el grafo G definido en (4.5) junto con el conjunto de medidas D, dado en (4.6), podemos denotar a la "red" asociada a una imagen como

$$N = \{\, G\,;\, D\,\}. \tag{4.7}$$

Sin pérdida de generalidad, consideremos inicialmente un píxel coloreado: sea $P_a \in V$ este píxel y su color $\mathrm{col}(P_a) = 0$. Sea α un valor (o umbral) fijado, los valores de los umbrales α serán discutidos más adelante. El proceso de coloreado binario de una red N dada en (4.7) sigue un esquema de propagación en cada componente conexa desde los píxeles coloreados hasta los píxeles adyacentes, finalizando cuando todos los píxeles tengan asignado un color. Dado un nodo coloreado $P_a \in V$, $\mathrm{col}(P_a) \in \{0, 1\}$, el color que se asigna a cualquier nodo adyacente $P_b \in V$ depende de la medida de disimilitud $d_{ab} \in D$ comparado con un valor establecido α:

$$\mathrm{col}(P_b) \;=\; \begin{cases} \mathrm{col}(P_a) & \text{si} \quad d_{ab} < \alpha \\ 1 - \mathrm{col}(P_a) & \text{si} \quad d_{ab} \geq \alpha \end{cases} \tag{4.8}$$

para todo $e_{ab} \in E$. Teniendo en cuenta lo anterior, dada una red N, como la definida por (4.7), introduciremos la familia de $K + 1$ valores (diferentes) de umbrales

$\{\alpha_0, \alpha_1, \ldots, \alpha_K\}$ verificando:

$$\alpha_0 > \overline{d} \geq \alpha_1 > \cdots > \alpha_{K-1} \geq \underline{d} > \alpha_K \tag{4.9}$$

Inicialmente, para $\alpha = \alpha_0$, sean $E^0 = E$ y $G^0 = (V, E^0) = G$, teniendo en cuenta que $\alpha_0 > \overline{d}$, cuando el proceso de coloreado binario (4.8) es aplicado, todos los píxeles serán igualmente coloreados $\mathrm{col}^0(P) = 0$ para todo $P \in V$, y esto define una partición trivial $\mathcal{P}^0 = \left\{C_1^0\right\}$ donde $C_1^0 = V$.
Siguiendo los pasos dados en el Capítulo 2 para el algoritmo Divide-and-Link; construimos, para $t \in \{1, \ldots, K\}$, la familia de grafos parciales y bosques soportes:

$$\left\{ G^t = (V, E^t) \right\}_{t=1}^{K} \qquad \left\{ F^t = (V, W^t) \right\}_{t=1}^{K}$$

respectivamente; donde $W^t \subset E^t$ para todo $t \in \{1, \ldots, K\}$. Más específicamente, para cualquier $t \in \{1, \ldots, K\}$, este proceso se divide en las siguientes partes:

1. **El grafo parcial G^t:**

   Dado el grafo parcial $G^{t-1} = \left(V, E^{t-1}\right)$ y su respectiva partición asociada, denotada por $\mathcal{P}^{t-1} = \left\{ C_1^{t-1}, C_2^{t-1}, \ldots, C_{s_{t-1}}^{t-1} \right\}$, definimos el siguiente grafo parcial $G^t = \left(V, E^t\right)$ donde E^t se definido como en (2.8).

2. **El bosque soporte F^t:**

   El arreglo de las aristas $e \in E^t$, en el algoritmo D&L, para la construcción del bosque soporte F^t, se realiza a través de dos medidas asociadas a las aristas: disimilitud d_e y afinidad (o similitud) l_e. Sin embargo, en problemas de segmentación donde sólo se cuenta con una medida de disimilitud d_e asociada a las artistas (como en el caso de imágenes digitales), es natural usar como medida de afinidad (o similitud), l_e, asociada a las aristas, el "inverso" de la disimilitud, es decir; $l_e = 1/d_e$. Así, para un α_t fijado, la forma de construir la estructura ordenada $L = E_d \cup E_l$ mostrada en (2.9), es ordenando de manera decreciente las disimilitudes de las aristas:

   $$d_{e_1} \geq \cdots \geq d_{e_{m_1}} \geq \alpha_t > d_{e_{m_1+1}} \geq d_{e_{m_1+2}} \geq \cdots \geq d_{e_{m_1+m_2}}. \tag{4.10}$$

   Note que, en (4.10), se cumple que $d_{e_1} \geq \cdots \geq d_{e_{m_1}} \geq \alpha_t$; además, para $h = m_2, \ldots, 2, 1$, se cumple que $d_{e_{m_1+h}} < \alpha_t$ y

   $$\frac{1}{d_{e_{m_1+m_2}}} \geq \cdots \geq \frac{1}{d_{e_{m_1+1}}} \qquad \text{es decir,} \qquad l_{e_{m_1+m_2}} \geq \cdots \geq l_{e_{m_1+1}}.$$

   Finalmente, haciendo a $n_1 = m_1$ y $n_2 = 1 + (m_2 - h)$, con $h = m_2, \ldots, 2, 1$, se obtiene la secuencia ordenada de aristas dada en (2.9), y así podemos construir el bosque soporte F^t según el enfoque del algoritmo D&L.

3. **La partición $\mathcal{P}^t$:**

   Un conjunto cualquiera C_i^{t-1}, con $i \in \{1, 2, \ldots, s_{t-1}\}$, es dividido en tantos subconjuntos C_j^t como componentes conexas tiene el subgrafo generado por

   $$V_{i,(0)}^{t-1} \equiv \left\{ P \in C_i^{t-1} \mid \mathrm{col}^t(P) = 0 \right\} \quad \text{y} \quad V_{i,(1)}^{t-1} \equiv \left\{ P \in C_i^{t-1} \mid \mathrm{col}^t(P) = 1 \right\}$$

   Todos estos conjuntos definen una partición $\mathcal{P}^t = \left\{ C_1^t, C_2^t, \ldots, C_{s_t}^t \right\}$.

Finalmente, la familia de $K+1$ umbrales, $\{\alpha_0, \alpha_1, \ldots, \alpha_K\}$, se definirán entonces una familia de particiones jerárquicas $\{\mathcal{P}^0, \mathcal{P}^1, \ldots, \mathcal{P}^K\}$ a través del proceso anterior.

4.3. Segmentación en Escala de grises

Con el fin de ilustrar el proceso de segmentación jerárquica para imágenes, vamos a utilizar un ejemplo de una imagen en tono de grises, re-escalada al intervalo $[1, 5]$, como se muestra a continuación:

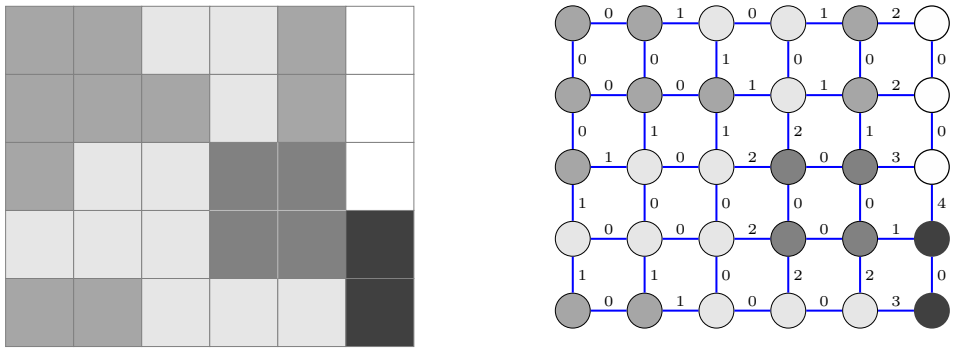
Ejemplo 4.1.
Considere la imagen en escala de grises que se muestra en la Figura 4.2a, cuya matriz de intensidad es:

$$P = \begin{bmatrix} 3 & 3 & 2 & 2 & 3 & 1 \\ 3 & 3 & 3 & 2 & 3 & 1 \\ 3 & 2 & 2 & 4 & 4 & 1 \\ 2 & 2 & 2 & 4 & 4 & 5 \\ 3 & 3 & 2 & 2 & 2 & 5 \end{bmatrix} \tag{4.11}$$

donde $P(i, j)$ en (4.11) representa una de intensidad, en escala de grises, tomando valores dentro del conjunto $\{1, 2, 3, 4, 5\}$. Consideremos la distancia entre píxeles definida por:

$$d\big((i, j), (i^*, j^*)\big) = \Big| P(i, j) - P(i^*, j^*) \Big| \tag{4.12}$$

para (i, j) e (i^*, j^*) en $[1, 2, \ldots, 5] \times [1, 2, \ldots, 6]$.



(a) (b)

Figura 4.2: (a): Imagen en escala de grises, y (b): red asociada N.

La Figura 4.2b muestra una representación de la red asociada a la imagen, con valores de píxeles (nodos) dados por (4.11), y cuyas distancias entre píxeles (aristas) se calculan usando (4.12). La analizar la Figura 4.2b, es claro que $\overline{d} = 4$ y $\underline{d} = 1$. Para este ejemplo, consideremos el parámetro $K = 6$, con un conjunto de umbrales $\{5, 4, 3, 2, 1, 0\}$; luego:

$$\alpha_0 = 5 > \overline{d} \geq \alpha_1 = 4 > \alpha_2 = 3 > \alpha_3 = 2 > \alpha_4 = 1 \geq \underline{d} > \alpha_5 = 0.$$

Excluyendo las particiones triviales $\mathcal{P}^0$ y $\mathcal{P}^5$, las $K-1=4$ particiones no triviales obtenidas para este ejemplo, son mostradas en la Figura 4.3. Se puede observar que $\big(s_0, s_1, \ldots, s_5\big) = (1, 2, 3, 8, 16, 30)$, además, se muestran, en cada iteración, los conjuntos C_i^t pertenecientes a cada partición (los grupos C_i^t se delimitan con un fondo azul).

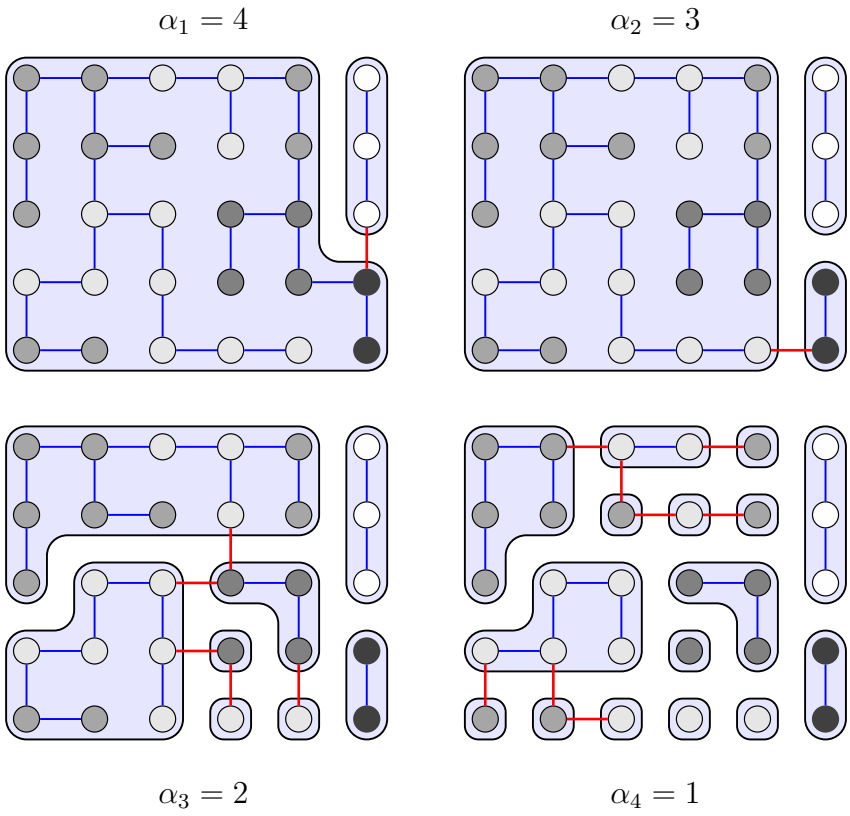


Figura 4.3: Segmentación jerárquica básica (no trivial)

4.4. Algoritmo D&L con Contracción de Nodos

A menudo, nos encontramos con redes que tienen regiones muy homogéneos y sería conveniente que permanecieran compactas para cualquier umbral utilizado en la segmentación. Estos grupos muy similares (o nodos adyacentes similares) son bastante comunes las imágenes, por lo que es importante tener en cuenta estos hechos en la aplicación de un proceso de segmentación. El proceso de segmentación jerárquica que hemos desarrollado en la Capítulo 2 puede ser mejorado siguiendo esta idea, como se muestra a continuación. Sea α_τ un umbral de similaridad dado, los extremos de cualquier arista con un peso inferior a este valor serán considerados como un único píxel y las aristas incidentes a este nuevo pixel son aquellas que eran incidentes a cualquiera de los píxeles de la arista inicial. Como consecuencia, los píxeles de estas aristas nunca serán separados en el proceso jerárquico.

En términos más generales, la operación anterior se puede realizar en un conjunto de aristas mediante la contracción de nodos (en cualquier orden): Sea

$$E(\alpha_\tau) \;=\; \big\{\, e_{ab} \in E \;\mid\; d_{ab} \leq \alpha_\tau \,\big\} \tag{4.13}$$

el conjunto de aristas con pesos inferiores a este valor; luego cualquier componente conexa del grafo parcial $G(\alpha_\tau) = \big(V, E(\alpha_\tau)\big)$ se fusiona en un nuevo vértice, donde las aristas incidentes a este nuevo vértice corresponderán a las aristas los píxeles de la componente conexa asociada a dichos píxeles. Este proceso de obtener grafos compactados es una técnica ampliamente utilizada en teoría de grafos (ver [51] para más detalles).

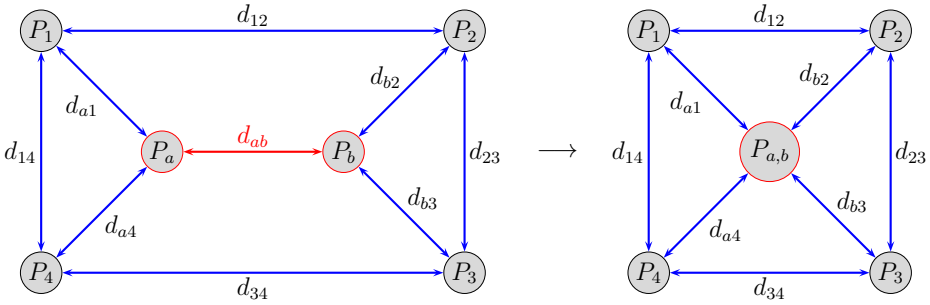


Figura 4.4: Contracción de nodos: $d_{ab} \leq \alpha_\tau$.

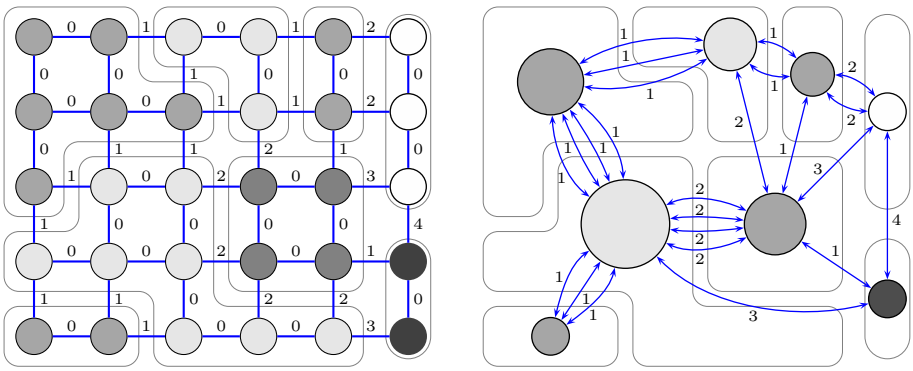
Definición 4.1 (Algoritmo jerárquico mejorado): *Considere la red* $N = \{G; D\}$ *dada en (4.7), dada una tolerancia* α_τ*, el algoritmo de segmentación jerárquica mejorado se define entonces como la aplicación del algoritmo básico jerárquico sobre la red* $\underline{N} = \big\{ \underline{G}; \underline{D} \big\}$*, con* $\underline{G} = (\underline{V}, \underline{E})$*, donde* $\underline{V}$ *es el conjunto de componentes conexas de* $G(\alpha_\tau)$*;* $\underline{E}$ *es el conjunto de aristas con pesos mayores que* α_τ*:*

$$\underline{E} \;=\; E \setminus E(\alpha_\tau) = \big\{ e_{ab} \in E \;\mid\; d_{ab} > \alpha_\tau \big\}, \tag{4.14}$$

y $\underline{D}$*, definido por (4.6), es aplicado sobre* $\underline{E}$ *, es decir;* $D = \{d_{ab} \;\mid\; e_{ab} \in \underline{E}\}$.

4.4.1. Algoritmo mejorado: escala de grises

Continuando con la red descrita en el Ejemplo 4.1, si utilizamos un umbral de similaridad (tolerancia) $\alpha_\tau = 0$; así la nueva red $\underline{N}$ tendrá una estructura como la mostrada en la Figura 4.5.



(a) (b)

Figura 4.5: Reducción del de la red: (a): Red N, donde $|V| = 30$ y $|E| = 49$; y (b): Red $\underline{N}$ donde $|\underline{V}| = 8$ y $|\underline{E}| = 24$. Los píxeles que cumplen que $d_{ab} \leq \alpha_\tau = 0$, están delimitados con bordes grises.

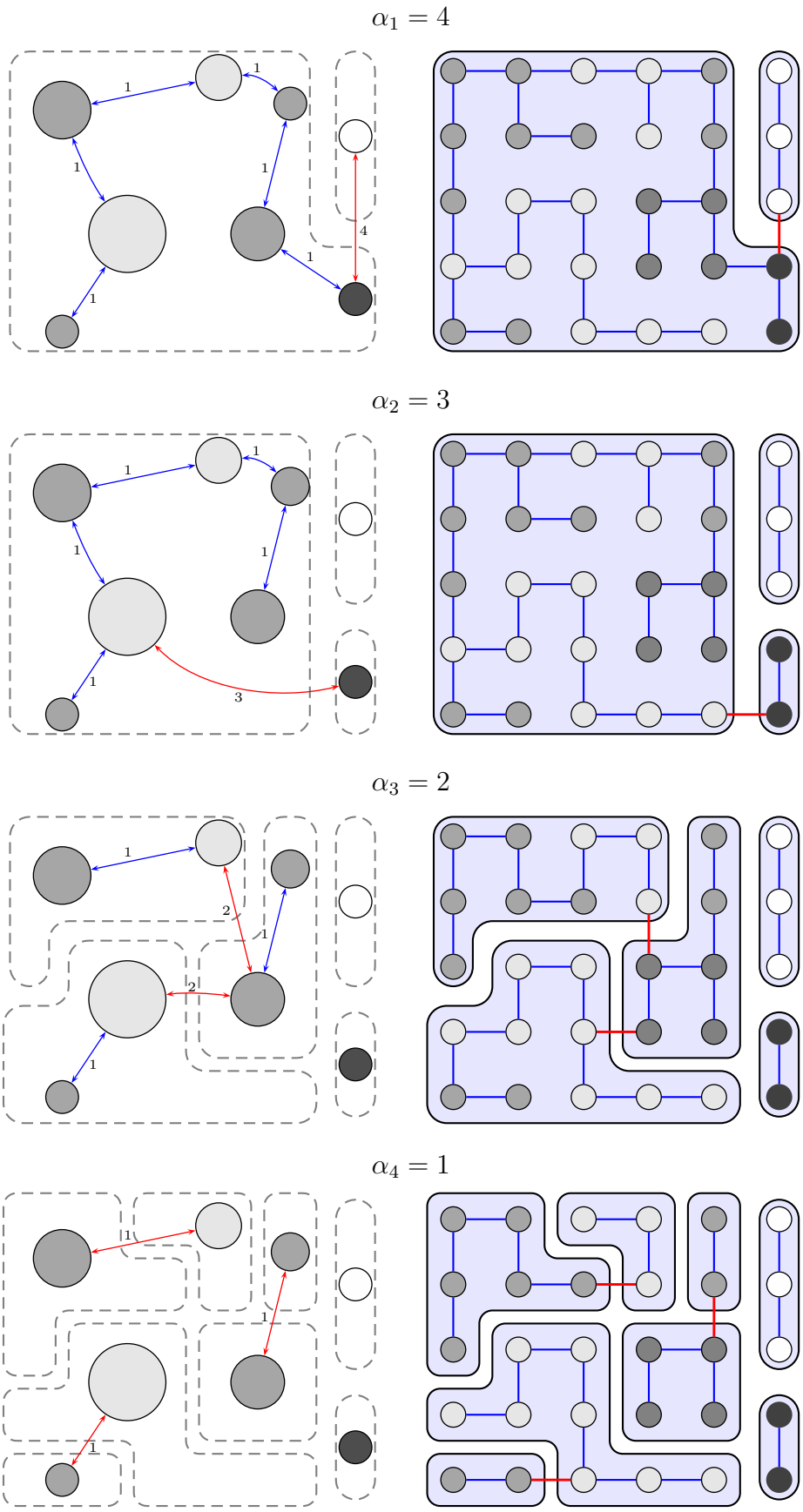


Figura 4.6: Algortimo jerárquico mejorado: las aristas rojas son las de corte

La sucesión de particiones no triviales del algoritmo jerárquico mejorado se muestran en la Figura 4.6. Note que $(s_0, s_1, \ldots, s_5) = (1, 2, 3, 5, 8, 30)$; las particiones finales se obtienen expandiendo el conjunto $\underline{V}$.

Observaciones:

- Luego de aplicar el proceso de segmentación jerárquica con contracción de nodos, se obtienen las particiones $\left\{ \underline{\mathcal{P}}^0, \underline{\mathcal{P}}^1, \ldots, \underline{\mathcal{P}}^K \right\}$. Este proceso de segmentación finaliza cuando los conjuntos $\underline{C}_i^t$ para cualquier i y t son "expandidos" de forma tal que el conjunto original de píxeles, V, es cubierto en su totalidad.
- El algoritmo jerárquico mejorado descrito en la Definición 4.1 es visto como una *generalización* del algoritmo jerárquico básico, puesto que valores del umbral (o tolerancia) que cumplen la desigualdad: $\alpha_\tau < \underline{d}$ producirán la misma partición entregada por el algoritmo básico (pues no habrán asociaciones de píxeles a priori).

4.5. Calidad de una Partición

Dado que estamos desarrollando técnicas de agrupación, lo usual es esperarse que los píxeles dentro de cada grupo sean más similares entre si que los píxeles asignados a grupos adyacentes. De esta manera, una propiedad deseable de una partición es que la distancia mínima entre dos grupos adyacentes sea mayor que la que existe entre los miembros de cada grupo, de lo contrario, estos grupos adyacentes similares deberían estar unidos (esta es la idea inicial de nuestro algoritmo jerárquico mejorado,).

Dada una partición $\mathcal{P}^t = \left\{ C_1^t, C_2^t, \ldots, C_{s_t}^t \right\}$, introducimos los siguientes índices:

$$d^t_{\text{within-}i} = d^t_{\text{w}_i} \equiv \text{máx} \left\{ d_{ab} \;\middle|\; a, b \in C_i^t \right\} \tag{4.15}$$

para todo $i \in \{1, 2, \ldots, s_t\}$, que mide el índice de disimilitud dentro (within) de cada agrupación i. Tomamos el mínimo de este índice en todos los grupos, es decir:

$$d^t_{\text{within}} = \overline{d^t_{\text{w}}} \equiv \frac{1}{s_t} \sum_{i=1}^{s_t} d^t_{\text{w}_i}, \tag{4.16}$$

así, el valor que entrega el índice (4.16) será asociado a la disimilitud dentro de los grupos de la partición $\mathcal{P}^t$. Por otra parte, introducimos otro índice que mide la disimilitud entre (between) los diferentes grupos adyacentes:

$$d^t_{\text{between}} = d^t_{\text{b}} \equiv \min_{e \in E(\mathcal{P}^t)} \left\{ d_e \right\} \tag{4.17}$$

donde

$$E(\mathcal{P}^t) \equiv \left\{ e_{ab} \in E \;\middle|\; a \in C_i^t \;;\; b \in C_j^t \quad , \quad i \neq j \right\}.$$

Una buena partición, $\mathcal{P}^t$, debe entonces verificar la desigualdad $d^t_{\text{b}} > \overline{d^t_{\text{w}}}$ en un sentido amplio, para indicar que las diferencias son mayores entre las agrupaciones adyacentes que dentro de los grupos. Los operadores dados en (4.15), (4.16) y (4.17) han sido introducido para medir la calidad de una partición en dicho sentido. En general una

gama más amplia de a operadores pudieran construirse con el objetivo de evaluar la calidad de una partición, y la forma de construirlos es explicada en la Subsección 4.5.1. En la Tabla 4.1 se muestra el comportamiento de los índices d_{w}^t y d_{b}^t dados por (4.16) y (4.17), para el Ejemplo 4.1 cuando es segmentado con el "Algoritmo Jerárquico Básico" y el "Algoritmo Jerárquico Mejorado".

Tabla 4.1: Calidad de las particiones. Comparación.

	t	α_t	s_t	$\{d_{\mathrm{w}_1}^t, \ldots, d_{\mathrm{w}_{s_t}}^t\}$	$\overline{d_{\mathrm{w}}^t}$	d_{b}^t
Algoritmo	1	**4**	2	{ 3,0 }	1.50	2
Jerárquico	2	**3**	3	{ 2,0,0 }	0.67	1
Básico	3	**2**	8	{ 1,0,1,0,0,0,0,0 }	0.25	0
	4	**1**	16	{ 0,0,0,. . . ,0 }	0.00	0
Algoritmo	1	**4**	2	{ 3,0 }	1.50	2
Jerárquico	2	**3**	3	{ 2,0,0 }	0.67	1
Mejorado	3	**2**	5	{ 1,1,0,1,0 }	0.60	1
	4	**1**	8	{ 0,0,0,0,0,0,0,0 }	0.00	1

De la Tabla 4.1 puede concluirse que el "Algoritmo Jerárquico Mejorado" verifica que $d_{\mathrm{b}}^t > \overline{d_{\mathrm{w}}^t}$ en todas sus particiones $\mathcal{P}^t$, para $t \in \{1, 2, 3, 4\}$.

4.5.1. Medidas Generales de Calidad de la Partición

Se puede decir que dos grupos son "adyacentes" si al menos un pixel de un grupo tiene un vecino en el otro. Formalmente, fijado un umbral α_t obtenemos una partición $\mathcal{P}^t = \left\{ C_1^t, C_2^t, \ldots, C_{s_t}^t \right\}$ a través de algún algoritmo de agrupación. Luego, se dice que los grupos $C_i^t, C_j^t \in \mathcal{P}^t$ son adyacentes si $\left|C_i^t, C_j^t\right| > 0$, con

$$\left|C_i^t, C_j^t\right| = \mathrm{card}\Big\{ e_{ab} \in E \;\Big|\; P_a \in C_i^t \;;\; P_b \in C_i^t,\; i \neq j \Big\} \tag{4.18}$$

donde "card" en (4.18) hace referencia a la carnalidad del conjunto. Note que, dependiendo del bosque soporte generado, pueden obtenerse particiones con grupos adyacentes muy similares o muy disimilares. Una forma de medir la calidad de la partición, para algún α_t dado, es a través de operadores de agregación (ver [42]), para ayudar a medir el grado de disimilaridad entre grupos adyacentes.

Sea $\mathcal{A}(C_i^t)$ algún operador de agregación definido en el conjunto de píxeles C_i^t (por ejemplo, la media, la mediana, máx, mín, entre otros),y sea $H^t : \mathcal{P}^t \times \mathcal{P}^t \to \mathbb{R}$ una función real-valuada definida como:

$$H^t\big(C_i^t, C_j^t\big) = \delta\Big(\mathcal{A}\big(C_i^t\big), \mathcal{A}\big(C_j^t\big)\Big) \tag{4.19}$$

donde δ es alguna función de distancia. A partir de (4.19) podemos definir un índice general de calidad, $H_{\min}^t$, que dependerá del operador $\mathcal{A}$ y la distancia δ, como:

$$H_{\min}^t = \min_{|C_i^t, C_j^t|>0} \Big\{ \delta\Big(\mathcal{A}\big(C_i^t\big), \mathcal{A}\big(C_j^t\big)\Big) \Big\}. \tag{4.20}$$

Note que valores grandes del índice (4.20) significa mayor disimilaridad entre los grupos de la partición $\mathcal{P}^t$. Si aplicamos índice de calidad de partición (4.20) en el Ejemplo 4.1, utilizando la media y la mediana como operadores de agregación, es decir; $\mathcal{A}(C_i^t) = \overline{C_i^t}$ y $\mathcal{A}(C_i^t) = \widetilde{C_i^t}$, respectivamente; y además usamos $\delta = \|\cdot\|$ la distancia Euclídea, entonces $H_{\min}^t$ en (4.20) puede escribirse, para cada caso, como:

$$\overline{H}_{\text{mín}}^t = \min_{|C_i^t, C_j^t|>0} \left\{ |\overline{C_i^t} - \overline{C_j^t}| \right\} \tag{4.21}$$

$$\widetilde{H}_{\text{mín}}^t = \min_{|C_i^t, C_j^t|>0} \left\{ |\widetilde{C_i^t} - \widetilde{C_j^t}| \right\} \tag{4.22}$$

y así, podemos comparar el Algoritmo jerárquico básico y el Algoritmo Jerárquico Mejorado con estos dos índices, como se muestra en la Tabla 4.2.

Tabla 4.2: Calidad de las particiones: Comparaciones en media y mediana

	Algoritmo Jerárquico Básico			Algoritmo Jerárquico Mejorado		
$\boldsymbol{\alpha}$	N	$\overline{H}_{\text{mín}}^t$	$\widetilde{H}_{\text{mín}}^t$	N	$\overline{H}_{\text{mín}}^t$	$\widetilde{H}_{\text{mín}}^t$
4	2	1.89	2.00	2	1.89	2.00
3	3	1.72	2.00	3	1.72	2.00
2	8	0.00	0.00	5	0.52	1.00
1	16	0.00	0.00	8	1.00	1.00

Note que, en el Algoritmo Jerárquico Básico, $\overline{H}_{\text{mín}}^t$ y $\widetilde{H}_{\text{mín}}^t$, definidos en (4.21) y (4.22), son cero para $\alpha = 2, 1$; esto significa que existen grupos adyacentes cuya diferencia es cero (menor calidad en la partición); sin embargo esto no ocurre en el Algoritmo Jerárquico Mejorado.

4.6. Experiencias Computacionales

En esta sección, se muestran algunas experiencias computacionales en las que aplicamos la segmentación jerárquica de algunas imágenes, en escala de grises y RGB-color, de la base de datos Berkeley `http://www.eecs.berkeley.edu/Research/ Projects/ CS/ vision/bsds/`. Las salidas, para un α_t dado, es una imagen en el que cada píxel está pintado de blanco, "1", si se trata de un borde de algún grupo, y negro, "0", en caso contrario. Inicialmente, tendremos una segmentación jerárquica usando la imagen en escala de grises mostrada en la Figura 4.7a, la cual tiene dimensiones de 481×321 píxeles, y cuyos valores se encuentran en el rango de escala de grises $[0, 255]$. Las Figuras $4.8b_1, \ldots, 4.8b_3$ muestran una segmentación jerárquica para una secuencia de umbrales elegidos automáticamente por el algoritmo (con el objetivo de la inclusión de un tamaño fijo de aristas de corte).

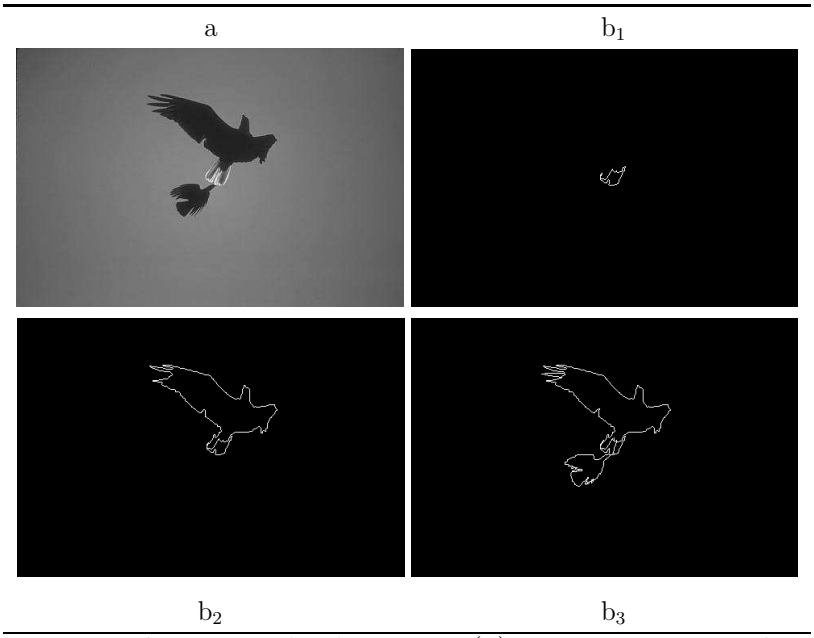


Figura 4.7: Segmentación en escala de grises. (a): imagen original, (b_1),. . .,(b_3): resultados usando la segmentación jerárquica

En la Figura 4.8, se muestra la segmentación de un paisaje en escala de grises, de dimensión 481×321 píxeles, usando cinco umbrales tomados automáticamente por el algoritmo de segmentación.

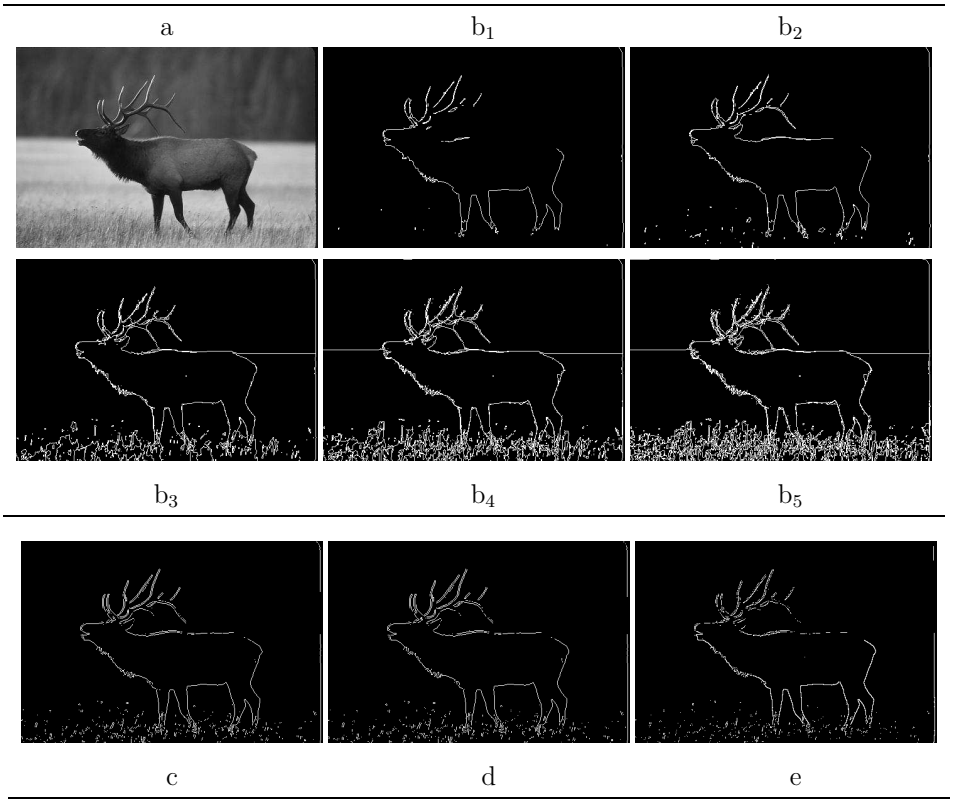


Figura 4.8: Segmentación en escala de grises. (a): imagen original, (b_1),. . .,(b_5): resultados usando la segmentación jerárquica propuesta, (c): algoritmo de Sobel, (d): algoritmo de Prewitt y (e): algoritmo de Roberts.

Las Figuras 4.8c, 4.8d and 4.8f muestran los resultados obtenidos con los enfoques de Sobel, Prewitt y Roberts, respectivamente; y que pueden ejecutarse directamente desde Matlab. Note que, la información que entregan las imágenes c, d o f se pueden obtener también en las imágenes b_i, para $i \in \{1, \ldots, 5\}$.

4.6.1. Clases de datos y tipos de imágenes

Inicialmente ilustraremos una tabla con los principales tipos de datos soportados por la mayoría de los software de programación y tratamiento de imágenes, en particular por Matlab, y el IPT (Image Processing Tools) para los valores de los pixeles representados.

Tabla 4.3: Tipos de valores para los píxeles de una imagen

Nombre	**Descripción**
`double`	Doble precisión, número de punto flotante en un rango aproximado $[-10^{308}, 10^{308}]$ (8 bytes por elemento)
`uint8`	Enteros positivos, 8-bit, en el rango [0,255] (1 byte por elemento)
`uint16`	Enteros positivos, 16-bit, en el rango [0,65535] (2 bytes por elemento)
`uint32`	Enteros positivos, 32-bit, en el rango [0,4294967295] (4 bytes por elemento)
`int8`	Enteros, 8-bit, en el rango [-128,127] (1 byte por elemento)
`int16`	Enteros, 16-bit, en el rango [-32768,32767] (2 bytes por elemento)
`int32`	Enteros 16-bit, en el rango [-2147483648,2147483647] (4 bytes por elemento)
`single`	Precisión simple, número de punto flotante en un rango aproximado $[-10^{38}, 10^{38}]$ (4 bytes por elemento)
`Char`	Caracteres (2 bytes por elemento)
`logical`	Valores de 0 o 1 (1 byte por elemento)

Los tipos de imágenes usualmente usados son de los siguientes tipos:

- **Imágenes de intensidad**: es una matriz de datos cuyos valores han sido escalados para representar intensidades. Así por ejemplo, cuando la matriz de datos es de clase `uint8` o `uint16`, ésta tiene valores enteros en los rangos [0,255] y [0,65535] respectivamente. Si la imagen es de clase `double`, así que los valores escalados para imágenes de intensidad de esta clase están en el rango [0,1] (por convención).

- **imágenes binarias**: en Matlab, una imagen binaria es un arreglo lógico de ceros y unos. Así, un arreglo de 0's y 1's cuyos valores pertenezcan a la clase, por ejemplo, `unit8`, no es considerada una imagen binaria en Matlab. Un arreglo numérico es convertido en binario usando la función `logical`. Así, por ejemplo, si `f` es un arreglo numérico que consiste de 0's y 1's, podemos crear un arreglo lógico, digamos `g` usando la sentencia: `g=logical(f)`. La forma de chequear si un arreglo, digamos `h`, es lógico; es a través de la función `islogical`, así: `islogical(h)`; en caso afirmativo, la salida es un 1; en caso negativo la salida es 0.

- **Imágenes RGB**: es un arreglo de dimensiones $M \times N \times 3$, de pixeles donde cada tripleta corresponde a las componentes Roja (R), verde (G) y azul (B) de una imagen en una localización espacial específica.

- **Imágenes indexadas**: estas imágenes tienen dos componentes: una matriz de datos (enteros), digamos `X`, y una matriz de mapa de colores, digamos `map`. La matriz `map` es un arreglo de dimensión $m \times 3$ de clase `double` que contiene valores de punto flotante en el rango [0,1]. La dimensión m en `map` es igual al número de colores definidos. Cada fila de `map` especifica las componentes rojo, verde y azul de un color en particular. Una imagen indexada usa un mapeo directo de los valores de intensidad de pixeles a valores del mapa de colores. El color de cada pixel es determinado usando el valor correspondiente de la matriz entera `X` como un puntero en `map`.

- **Imágenes FITS:** (Flexible Image Transport System) El sistema de transporte flexible FITS es un formato de archivo de datos estándar creado para conjuntos de datos astronómicos, aunque, dadas las posibilidades de este formato, su uso se ha expandido a conjuntos de datos más complejos.

Imágenes RGB

En esta parte discutiremos más a fondo la estructura y propiedades de las imágenes RGB (red, green, blue), pues es uno de los formatos más frecuentes en imágenes digitales. Como se dijo antes, una imagen RGB es un arreglo $M \times N \times 3$ de colores de los pixeles, donde, por convención, las tres imágenes que conjuntamente forman una imagen RGB están referenciadas a las componentes roja, verde y azul de la imagen, como lo muestra la Figura 4.9.

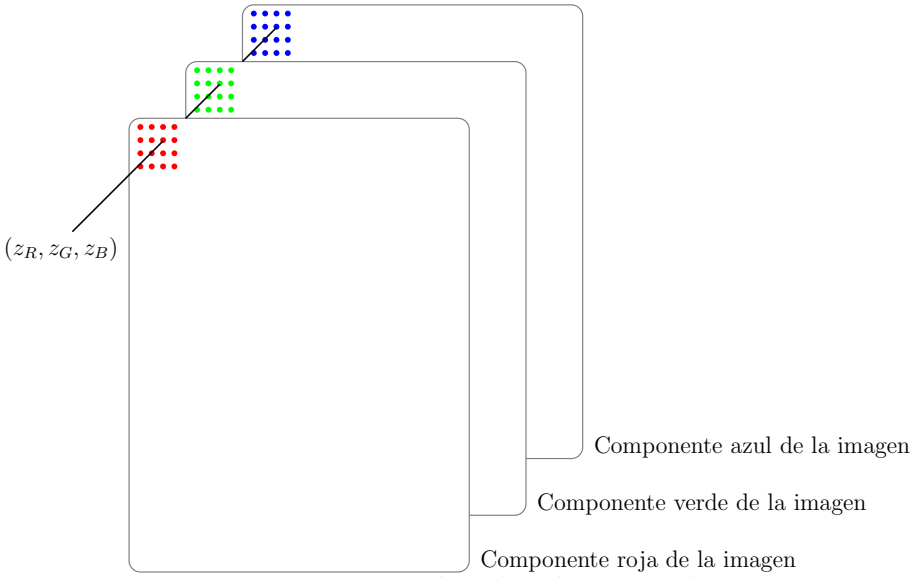


Figura 4.9: Tres componentes de color de un pixel

El tipo de datos de las componentes de la imagen determinan su rango de valores. Así, por ejemplo, si una imagen RGB es de clase `double`, el rango de valores es $[0, 1]$, similarmente, el rango sería $[0, 225]$ o $[0, 65525]$ si la imagen RGB es de clase `uint8` o `uint16`, respectivamente. El número de bits usados para representar los valores de los pixeles de las componentes de la imagen determina la 'profundidad de bits' (bit depth) de una imagen RGB. Así por ejemplo, si cada componente de una imagen es de 8 bits, entonces la imagen RGB correspondiente tendrá una profundidad de 24 bits.

Generalmente, el número de bits en las componentes es el mismo. En este caso, el número posible de colores en una imagen RGB es $(2^b)^3$ donde b es el número de bits en cada imagen componente. Para el caso de 8 bits, el número es 16,777.216 colores.

El comando `cat` permite concatenar tres componentes de imagen para que el resultado sea una imagen RGB. Su sintaxis es: `rgb_image`= cat(3, fR, fG, fB). El resultado de dicha sentencia es 'apilar' las tres imágenes `fR`, `fG` y `fB` en una sola (que es una forma de generar imágenes RGB).

Existen muchos desarrollos adicionales con imágenes de uso frecuente, como son por ejemplo, la conversión a otros espacios de color (NTSC, YCbCr, HSV, CMY, CMYK, HSI, entre otros). Sin embargo, el propósito de este capítulo es desarrollar los conceptos y las herramientas básicas para el procesamiento de imágenes, en particular, la segmentación de imágenes; en donde nos centraremos en imágenes RGB. Un estudio más amplio de la manipulación de imágenes usando el IPT de Matlab puede estudiarse en [52].

4.6.2. Aproximación a un espacio de medida uniforme

Con el fin de obtener una buena medida de color, se hace necesario trasladar una imagen RGB, a otro espacio de color más uniforme (por ejemplo, los espacios de color CIELAB o CIELUV; ver [81] para más información). Sean $P_1^{\text{rgb}} = (r_1, g_1, b_1)$ y $P_2^{\text{rgb}} = (r_2, g_2, b_2)$ dos colores en el espacio RGB; su representación en el espacio CIELAB viene dada por: $P_1^{\text{lab}} = (L_1, a_1, b_1)$ y $P_2^{\text{lab}} = (L_2, a_2, b_2)$ respectivamente. Así, una forma de medir la distancia entre P_1^{lab} y P_2^{lab} es

$$d_{1,2}^{\text{lab}} = \sqrt{(\Delta L)^2 + (\Delta a)^2 + (\Delta b)^2} \tag{4.23}$$

donde $\Delta L = L_2 - L_1$, $\Delta a = a_2 - a_1$ and $\Delta b = b_2 - b_1$. La distancia dada en (4.23) es conocida como CIE76, y es una de las definiciones de distancia entre colores más ampliamente usadas, dada por la Comisión Internacional de Iluminación CIE (International Commission on Illumination) en 1976.

La definición de 1976 se amplió para hacer frente a una cierta falta de uniformidad de percepción, al tiempo que conserva el sistema "Lab" del espacio de color. Esta modificación se conoce como CIE94, y viene dada por:

$$d_{1,2}^{\text{CIE94}} = \sqrt{\left(\frac{\Delta L}{k_L S_L}\right)^2 + \left(\frac{\Delta C}{k_C S_C}\right)^2 + \left(\frac{\Delta H}{k_H S_H}\right)^2} \tag{4.24}$$

donde $\Delta L = L_2 - L_1$, $\Delta C = C_2 - C_1$ con $C_1 = \sqrt{a_1^2 + b_1^2}$ y $C_2 = \sqrt{a_2^2 + b_2^2}$; $\Delta H = \sqrt{\Delta a^2 + \Delta b^2 - \Delta C^2}$ con $\Delta a = a_2 - a_1$ y $\Delta b = b_2 - b_1$; $S_L = 1$, $S_C = 1 + K_1 C_1$, $S_H = 1 + K_2 C_1$ y donde k_L, k_c, k_H, K_1 y K_2 dependen de la aplicación: por ejemplo, en textiles, $[k_L, k_c, k_H, K_1, K_2] = [2, 1, 1, 0.048, 0.014]$; en artes gráficas $[k_L, k_c, k_H, K_1, K_2] = [1, 1, 1, 0.045, 0.015]$, etc.

En 2000, la CIE propuso una nueva fórmula, conocida como CIEDE2000, la cual usa datos adicionales a los de CIE94. Su ecuación mantiene la estructura base de CIE94, además de unos cuantos términos adicionales, incluyendo un producto cruzado. Su ecuación es:

$$d_{1,2}^{\text{CIE00}} = \sqrt{\left(\frac{\Delta L'}{k_L S_L}\right)^2 + \left(\frac{\Delta C'}{k_C S_C}\right)^2 + \left(\frac{\Delta H'}{k_H S_H}\right)^2 + R_T \left(\frac{\Delta C'}{k_C S_C}\right)\left(\frac{\Delta H'}{k_H S_H}\right)} \tag{4.25}$$

con

$$
\begin{aligned}
L' &= L, \quad \text{además } \overline{L'} = \frac{L_1 + L_2}{2}, \\
b'_i &= b_i \quad \text{para } i = 1, 2\ , \\
a'_i &= a_i + \frac{a_i}{2}\left(1 - \frac{\overline{C}^7}{\overline{C}^7 + 25^7}\right) \quad \text{para } i = 1, 2 \quad \text{y} \quad \overline{C} = \frac{C_1 + C_2}{2}, \\
C' &= \frac{C'_1 + C'_2}{2} \ \text{para} \quad C'_i = \sqrt{a_i'^2 + b_i'^2} \ \text{con } i = 1, 2, \text{ además } \ \Delta C' = C'_2 - C'_1: \\
h'_i &= \tan^{-1}\left(\frac{b'_i}{a'_i}\right) \text{ para } i = 1, 2, \text{ además } \overline{h'} = \frac{h'_1 + h'_2}{2} \\
T &= 1\text{-}0.17\cos(\overline{h'} - 30°) + 0.24\cos(2\overline{h'}) + 0.32\cos(3\overline{h'} + 6°)\text{-}0.20\cos(4\overline{h'} - 63°), \\
R_T &= -2\sqrt{\frac{\overline{C'}^7}{\overline{C'}^7 + 25^7}}\ \sin\left(60° \exp\left\{\frac{\overline{h'} - 275°}{25°}\right\}^2\right), \\
\Delta H' &= 2\sqrt{C'_1 C'_2}\sin\left(\frac{\Delta h'}{2}\right); \\
S_L &= 1 + \frac{0{,}015(\overline{L'} - 50)^2}{\sqrt{20 + (\overline{L'} - 50)^2}},\ S_C = 1 + 0{,}0045\overline{C'} \text{ y } S_H = 1 + 0{,}0015\overline{C'}T.
\end{aligned}
$$

En la actualidad CIE76 todavía se utiliza en diversas aplicaciones, aunque se tengan varias versiones con modificaciones para (4.23) como son CIE94 y CIEDE2000 entre otras. Sin embargo, para nuestros propósitos utilizaremos a CIE76 como una buena aproximación para la medida entre colores.

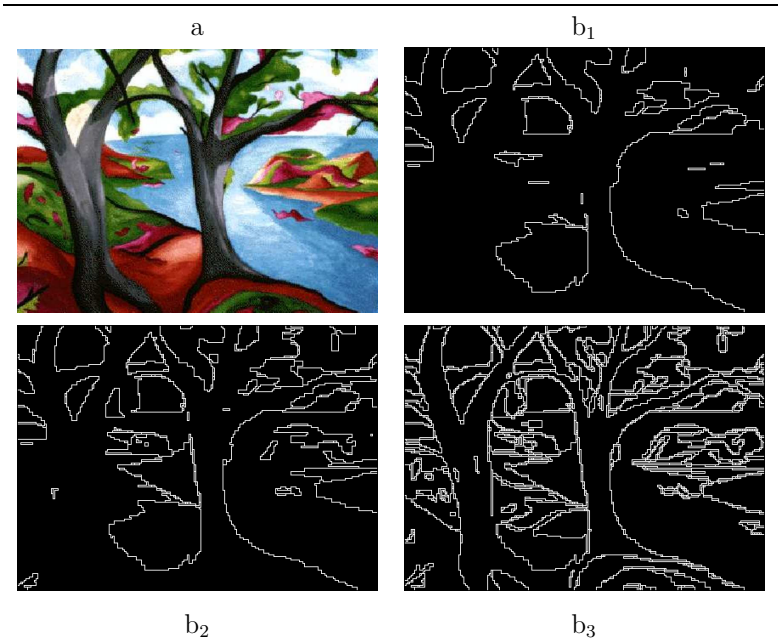


Figura 4.10: (a): Imagen original, b_1,...,b_3: segmentación jerárquica usando CIE76.

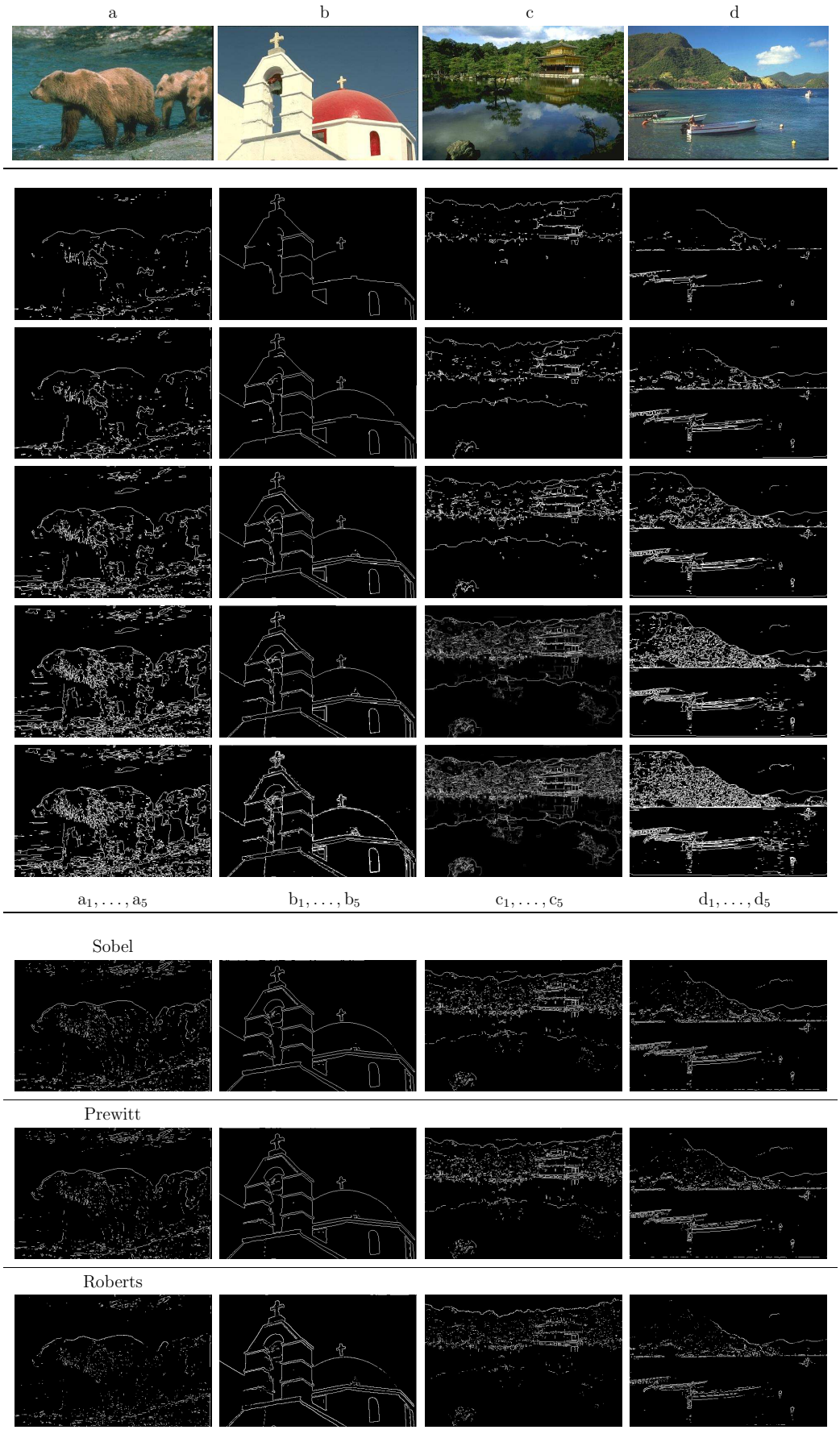


Figura 4.11: a,b,c y d: Imágenes originales. k_1,. . .,k_5: segmentación jerárquica con CIE76, y k=a,b,c,d. También se muestran los resultados de Sobel, Prewitt y Roberts.

4.7. Algunas Mejoras Computacionales

4.7.1. Redes Gruesas

En segmentación de imágenes basada en grafos, es una buena idea tratar de utilizar redes gruesas con el fin de reducir el tiempo de cálculo. La teoría de operadores de agregación puede ser usada con este propósito. En general, considere una imagen I con una red asociada $N = \{G = (V, E); D\}$, dada en (4.7). Sea $\Omega = \bigcup_{\lambda}^{n_\lambda} Q^\lambda$ un cubrimiento disjunto para la V donde los Q^λ son conjuntos de píxeles vecinos (conexos). Ahora, para algún operador de agregación $\mathcal{A}$, construimos la "red gruesa"

$$\hat{N} = \left\{ \hat{G} = (\hat{V}, \hat{E}) \ ; \ \hat{D} \right\}$$

donde los nodos, las aristas y el conjunto de distancias vienen dados por:

$$\hat{V} = \left\{ P^\lambda \ \middle| \ P^\lambda = \mathcal{A}(Q^\lambda); \quad \lambda = 1, 2, \ldots, n_\lambda \right\} \tag{4.26}$$

$$\hat{E} = \left\{ e_{\lambda,\lambda^*} = (P^\lambda, P^{\lambda^*}) \ \middle| \ P^\lambda, P^{\lambda^*} \in \hat{V} \right\} \tag{4.27}$$

$$\hat{D} = \left\{ d_{\lambda,\lambda^*} \ \middle| \ e_{\lambda,\lambda^*} \in \hat{E} \right\} \tag{4.28}$$

Ejemplo 4.2.
Considere una red N de $|V| = 24$ píxeles y una topología de cuatro vecinos, como se muestra en la Figura 4.12a (note que en red N tienen $|E| = 30$ aristas); además se muestra un cubrimiento $\Omega = Q^1 \cup \cdots \cup Q^6$ donde cada Q^λ agrupa cuatro píxeles. Ahora bien, a partir de dicha agrupación, en la Figura 4.12b se muestra la red gruesa $\hat{N}$ con $|\hat{V}| = 6$ y $|\hat{E}| = 7$, con $P^i = \mathcal{A}(Q^i)$ para algún operador de agregación $\mathcal{A}$, con $i = 1, \ldots, 6$.

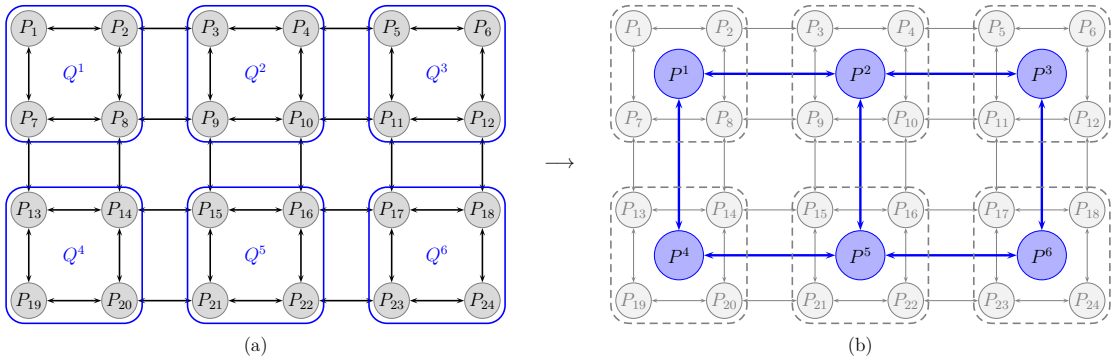


Figura 4.12: Redes gruesas. Construcción

Note que, en términos computacionales, es más rápido analizar la imagen manipulando la red $\hat{N}$ que la red inicial N; además, en muchos problemas, los resultados al hacer agrupación usando alguna de las dos redes es bastante similar, como lo veremos más adelante.

La Figura (4.13) muestra algunos ejemplos de cubrimientos básicos y sus respectivas redes asociadas:

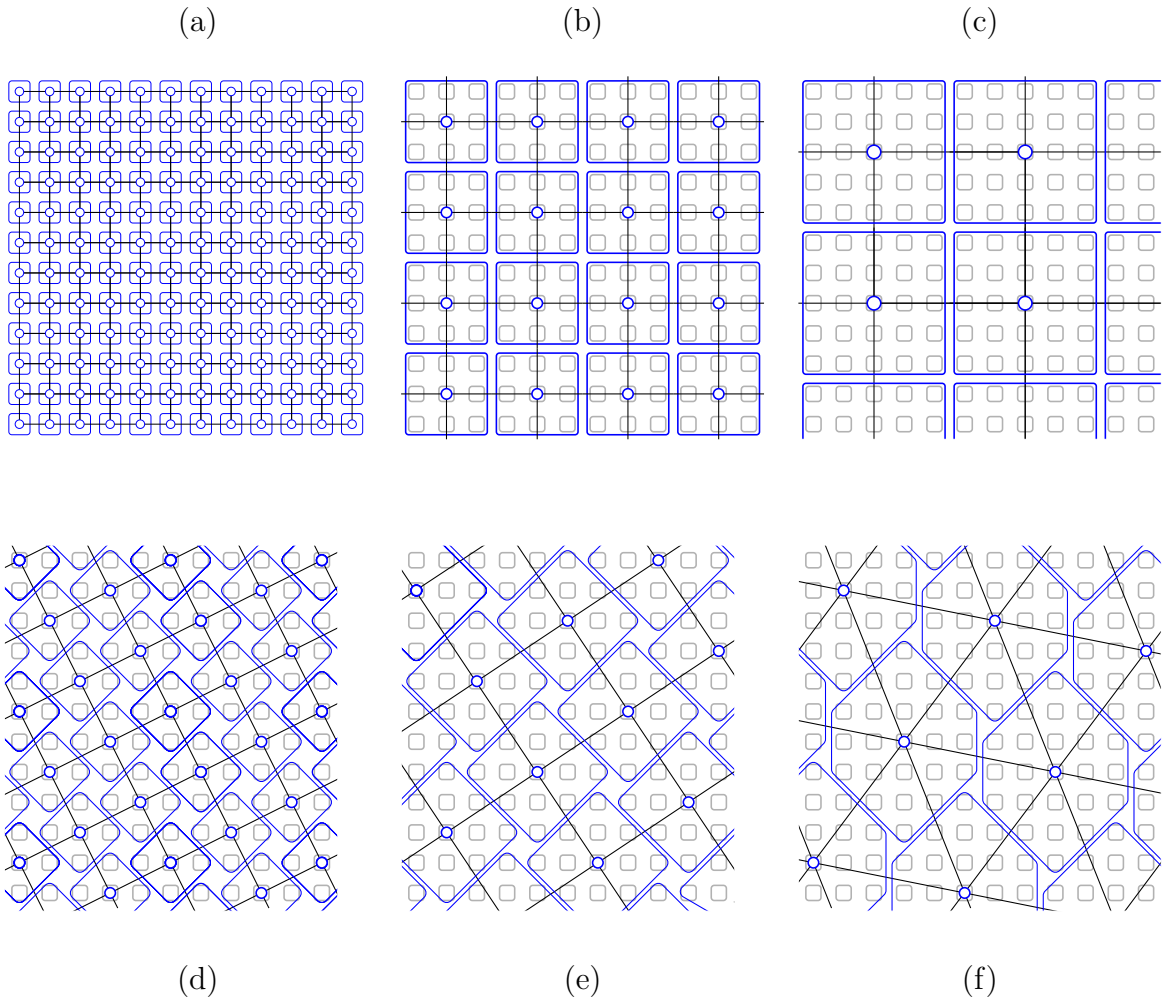


Figura 4.13: Cubrimientos simétricos: (a), (b) y (c): cuadrados, $Q^\lambda = \square_n$, con $n = 1, 9$ y 25 píxeles respectivamente. (d) y (e): diamante, $Q^\lambda = \diamond_n$, con $n = 5$ y 13 píxeles, respectivamente; y (f): hexagonal, $Q^\lambda = \hexagon_n$ con $n = 23$ píxeles.

A continuación mostramos la segmentación jerárquica de varias imágenes RGB, usando redes gruesas a partir de los cubrimientos $C^\lambda = \square_n$, con $n = 4, 9$ y 16 píxeles.

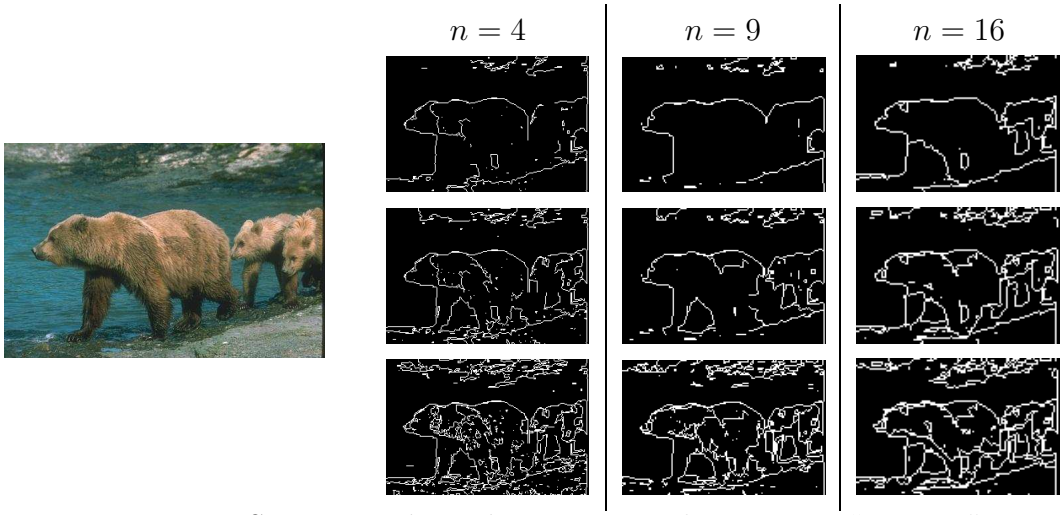


Figura 4.14: Segmentación jerárquica con redes gruesas. “osos.jpg”.

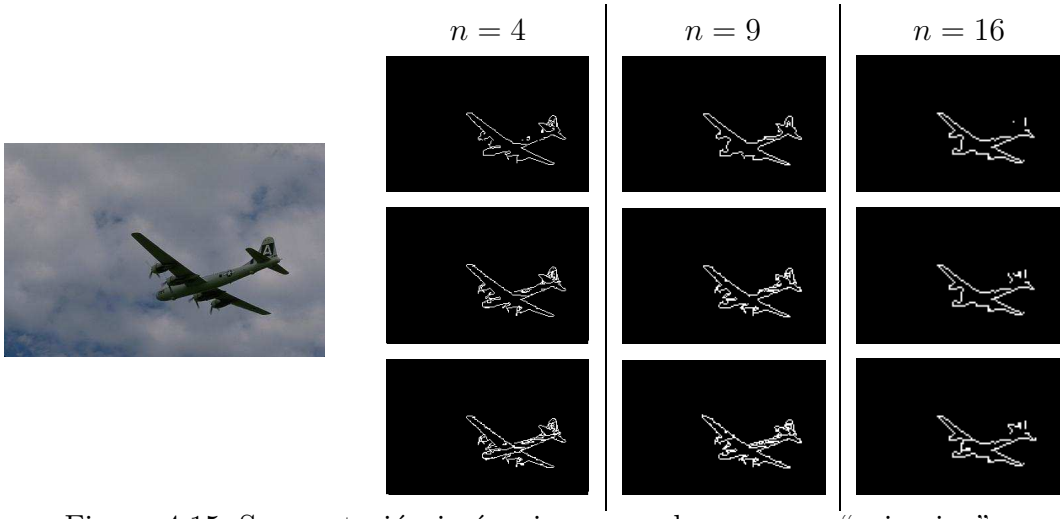


Figura 4.15: Segmentación jerárquica con redes gruesas. "avion.jpg".

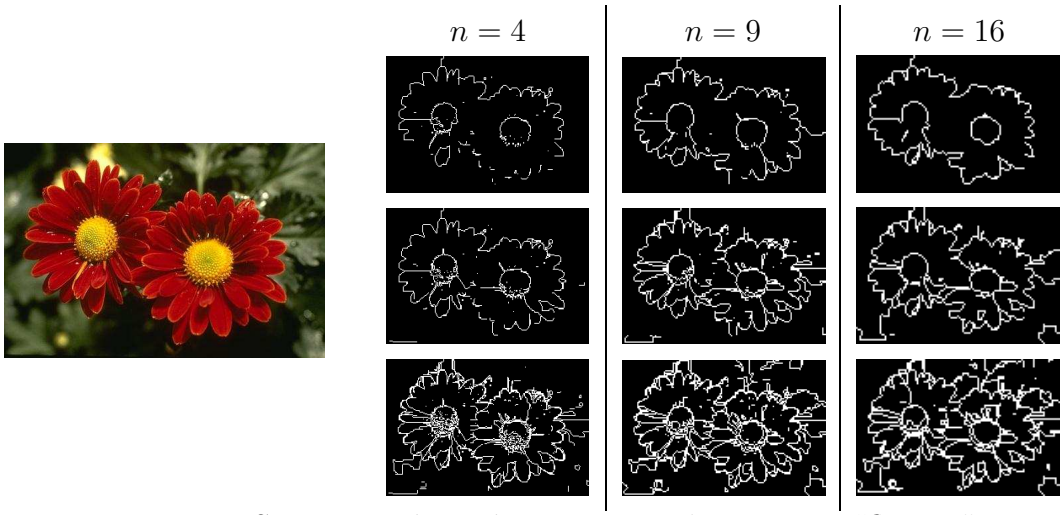


Figura 4.16: Segmentación jerárquica con redes gruesas. "flor.jpg".

Note que, en las Figuras 4.14, 4.15 y 4.16, la información general dada por las secuencias creadas a partir de las diferentes redes es similar bajo el algoritmo D&L; sin embargo, los tiempos de ejecución del algoritmo varía considerablemente, como lo muestra la Tabla 4.4.

Tabla 4.4: Resultados usando cubrimientos cuadrados, $Q^{\lambda} = \square_n$, con $n = 4, 9$ y 16 píxeles (resultados por segmentación)

	Tiempo CPU (seg)			
n	Oso	Avión	Flores	Promedio
1	12.5	8.10	11.20	10.60
4	1.10	0.90	1.00	1.00
9	0.41	0.32	0.38	0.37
16	0.20	0.16	0.18	0.18

4.8. Imágenes Astronómicas

Las imágenes astronómicas de cuerpos extensos proporcionan información crucial sobre las propiedades del medio interestelar. Sin embargo, estas imágenes tienen un fondo débil difícil de identificar o tratar. Así pues, las estructuras suelen ser identificadas después de una inspección visual.

Los algoritmos tratados en este trabajo permiten obtener una buena solución para identificar estructuras (incluso estructuras débiles), de una manera no supervisada (necesaria en bases de datos de imágenes astronómicas de terabites de información)

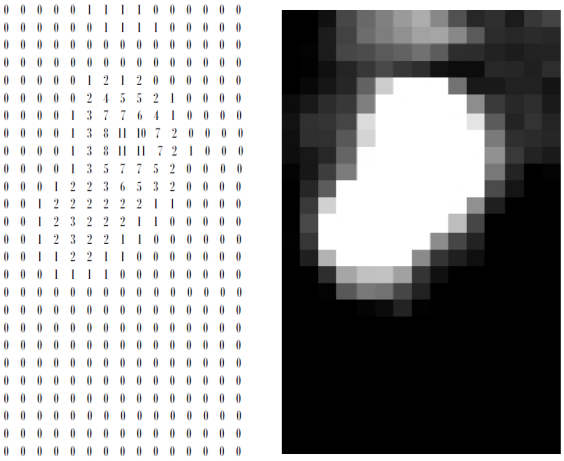
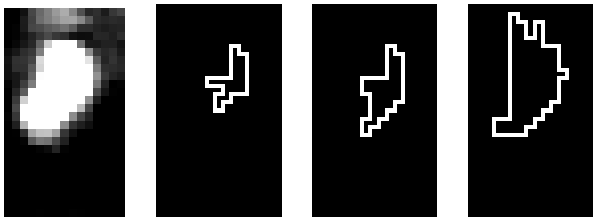


Figura 4.17: Imagen FITS

Las siguientes imágenes pertenecen un subconjunto de imágenes mayores las cuales han sido obtenidas de la base de imágenes astronómicas GALEX (Science Archive of the Galaxy Evolution Explorer)

4.8.1. Segmentación de Imágenes astronómicas

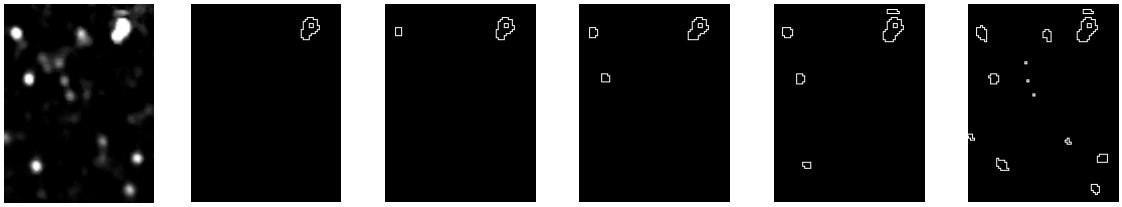
El algoritmo D&L permiten obtener una buena solución para identificar las estructuras débiles, de una manera no supervisada (necesario en bases de datos de imágenes astronómicas con Terabytes de información). En este sentido, cabe también destacar que el algoritmo aquí presentado podría ser adecuado para identificar estructuras de esa medida sobre varias imágenes sin necesidad de usar procedimientos de normalización, necesarios en el tratamiento de este tipo de imágenes.



El algoritmo aquí desarrollado, tiene en cuenta la borrosidad asociada a las características del fondo, así como la dificultad para evaluar si los fotones detectados en niveles de señal bajos son causadas por el ruido (ruido fotónico, ruido electrónico) o representan alguna señal real. A modo de ejemplo se analiza una imagen astronómica donde $\underline{\alpha} \equiv 0$ y $\overline{\alpha} \equiv 4{,}7$. Si elegimos, por ejemplo, la siguiente secuencia de umbrales $\alpha \in \{0{,}6,\ 0{,}8,\ 1,\ 1{,}3,\ 1{,}6\}$; la segmentación jerárquica se muestra en la Figura 4.18.

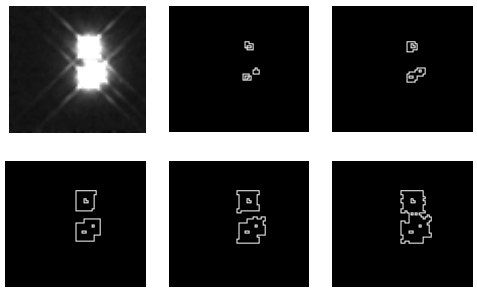
Veamos ahora el análisis de la siguiente imágen astronómica dada por la Figura 4.19, donde $\underline{\alpha} \equiv 0$ and $\overline{\alpha} \equiv 202374840$.

Ambas imágenes son subconjuntos de imágenes mayores que han sido descargadas del *Science Archive of the Galaxy Evolution Explorer* (GALEX) ver [72] para más detalles. GALEX es un telescopio espacial que observa el rango ultravioleta del espectro



a. b. c. d. e. f.

Figura 4.18: a: imagen original; b,c,d,e,f: segmentación jerarquizada para α= 1.6, 1.3, 1, 0.8 y 0.6 respectivamente.



a. b. c.

d. e. f.

Figura 4.19: a: imagen original; b,c,d,e,f: segmentación jerarquizada para $\alpha = [80,\ 50,\ 10,\ 1,\ 0{,}5] \times 10^6$ respectivamente.

electromagnético. La imagen original cubre un campo circular de visión de 0,6 grados de radio con centro en Ascensión Recta 4h34m8.s0 y Declinación 23d38'33"en la Nube Estelar de Tauro, una región cercana a la formación de estrellas. Se buscan filamentos tenues con el algoritmo. Además, los detectores de GALEX son detectores de conteo de fotones. Por este motivo, no hay lectura de ruido electrónico; el ruido en las imágenes puede ser simplemente modelado por las estadísticas de los fotones.

Las imágenes segmentadas muestran claramente cómo los objetos reales son claramente identificados. Además, la clasificación jerárquica aquí desarrollada permite obtener una graduación de borde y de los objetos, lo que da una visualización más realista de la situación.

4.9. Regiones Débiles en una Imagen Astronómica

El problema de detectar regiones débiles en una imagen astronómica es uno de los objetivos de "Observatorio Espacial Mundial - Ultravioleta" (WSO-UV, sigla en inglés); el cual es una misión espacial diseñada para estudiar el Cosmos en la región del espectro electromagnético ultravioleta. El WSO-UV será diez veces más sensible que el "Hubble Space Telescope". El WSO-UV permitirá realizar observaciones de extraordinaria importancia para el avance de muchos campos de la astrofísica. La misión tiene cinco objetivos científicos fundamentales: el estudio de la formación de las galaxias y de la evolución química del Universo, la medida de las propiedades de la materia difusa en el Universo y su distribución en los halos galácticos, la formación y evolución de la Vía Láctea, el papel de los discos en los motores astronómicos y la composición química y

las propiedades de las atmósferas de los planetas extrasolares gigantes.

4.9.1. D&L en detección de regiones débiles

A continuación mostraremos algunas experiencias computacionales de nuestro algoritmo Divede-and-Link adaptado para la detección de regiones débiles. Para ello, hemos utilizado en las imágenes un cubrimiento simétrico (como los mostrados en la Figura 4.13), $Q^{\lambda} = \square_n$, con $n = 8$, de forma tal que se puedan identificar grandes estructuras en las imágenes astronómicas. También se uso un $\alpha_\tau = 0$ para la contracción de nodos.

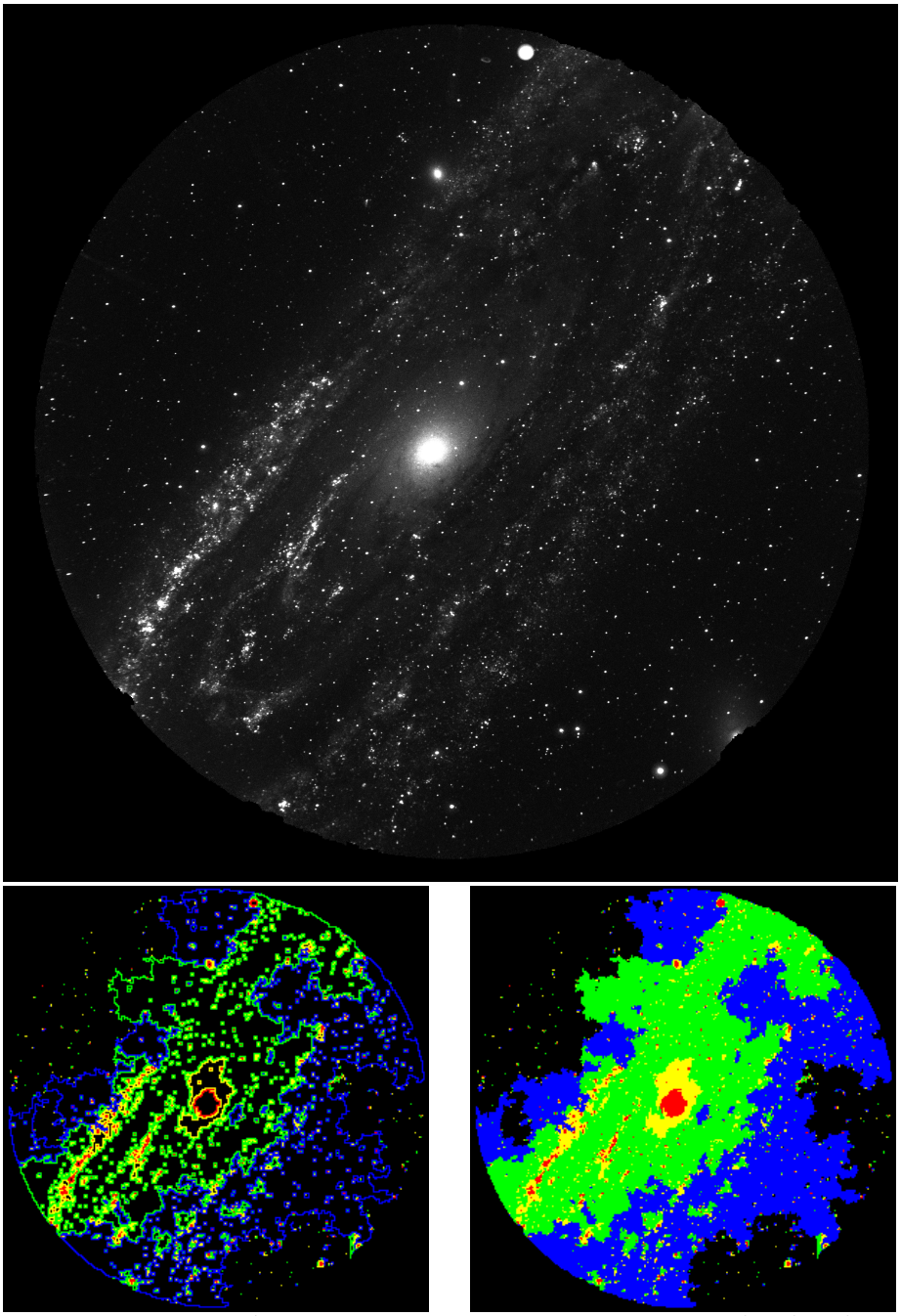


Figura 4.20: Imagen astronómica `NGA-M31-F1-nd-int.fits`, y los bordes y regiones débiles detectadas (regiones en azul))

Si usamos bordes de intensidad (de más intenso (rojo) a menos intenso (azul)), el proceso

jerárquico D&L nos permite ver los lugares donde se encuentran las regiones débiles, como se muestra en las Figuras 4.20 y 4.21.

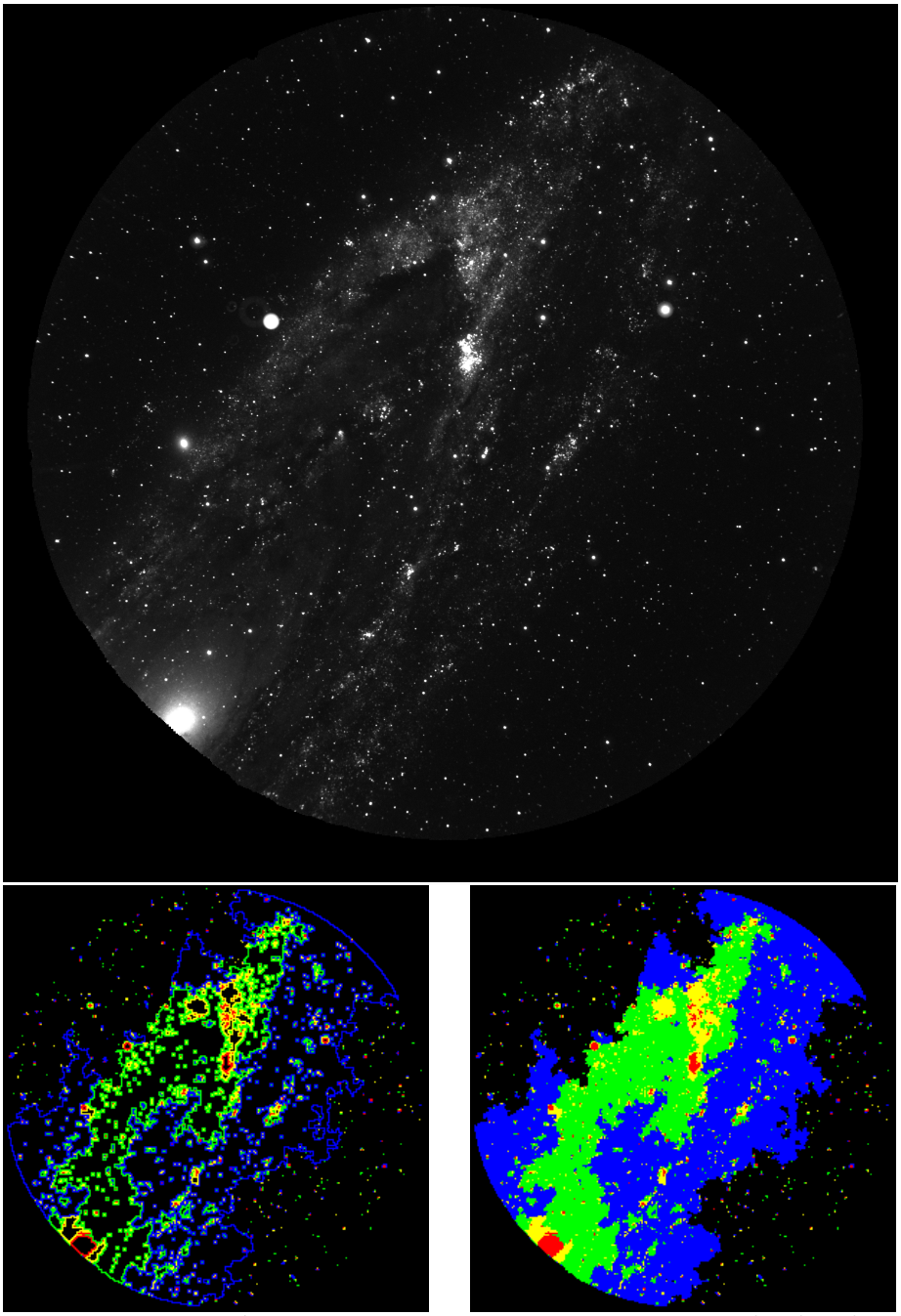


Figura 4.21: Imagen astronómica `NGA-M31-MOS0-nd-int.fits`, y los bordes y regiones débiles detectadas (regiones en azul).

Una ventaja ventaja del uso del algoritmo D&L en este tipo de problemas en imágenes astronómicas es que permite calibrarse dependiendo de los objetivos de la investigación, como en este caso, la búsqueda de regiones débiles.

Capítulo 5

Conclusiones y Líneas Futuras de Investigación

Contenido

5.1. Algoritmo Divide-and-Link

Los problemas de detección de comunidades (véase, por ejemplo, [37, 97]) se definen usualmente como un problema de agrupación en redes, en la que el objetivo principal es obtener una *buena* partición de la red. En este trabajo hemos abordado un problema más complejo: el *problema de agrupación jerárquica de la red* (HCNP en inglés), que persigue una *buena* partición jerárquica en la red. En este "nuevo problema" se da más relevancia al proceso dinámico de agrupación. Consideramos que muchas veces en el análisis de redes sociales, las particiones jerárquicas son más informativas que una única partición, debido a que se muestra la evolución de cómo se forman los grupos (aglomeración) o división (división), a partir de la situación inicial hasta el último paso. Como lo señala [22], *la jerarquización es un principio central de organización de redes complejas, y que tiene la capacidad de ofrecer una idea de muchos fenómenos de la red.* Un análisis jerárquico de la red representa una "salida" mucho más informativa que una "salida" no jerárquica.

Dado el HCNP, en este trabajo se ha propuesto un nuevo algoritmo para redes que permite agrupar un conjunto de elementos relacionados, a través de un grafo va-

lorado. Nuestro algoritmo Divide-and-Link reúne métodos jerárquicos de aglomeración y división, debido a que se basa en la construcción de bosques soportes, a través de dos enfoques complementarios: un primer estado de "división" y un segundo estado de "unión" de nodos. El núcleo del algoritmo propuesto es un procedimiento binario iterativo para grafos, basado en enfoques ya introducidos en [48, 49]. Este proceso base se introdujo para obtener una segmentación de una imagen digital modelada como una red de píxeles particular (véase [47, 45]). Sin embargo, en este nuevo algoritmo propuesto,el algoritmo Divide-and-Link, no hay limitaciones sobre la red, lo que permite aplicarlo a redes arbitrarias.

Otra característica importante del algoritmo Divide-and-Link es su complejidad polinomial. Este hecho, junto con la representación con dendogramas propuesta, permite el análisis visual de redes grandes. Estas representaciones visuales se vuelven de crucial importancia cuando nos enfrentamos a problemas complejos (ver, por ejemplo, [61]) o cuando la información está sujeta a alguna estructura subyacente (véase, por ejemplo, [30]).

Por otra parte, en el Capítulo 3, hemos adaptado nuestro algoritmo D&L (y su versión rápida D&LF), para hacer frente al problema de detección de comunidades donde la única información disponible es el grafo de la red. En esta adaptación se han utilizado dos criterios: la medida de intermediación "betweenness SP", y la medida de afinidad para nodos adyacentes definida en [60]. También hemos comparado nuestro algoritmo Divide-and-Link, a través de algunos "ejemplos de prueba" bien conocidos en la detección de comunidades, con los algoritmos jerárquicos más eficaces. Los algoritmos propuestos han mostrado buenos resultados en términos de modularidad.

Finalmente, nos gustaría hacer hincapié (véase también a [46]) en la necesidad de metodologías alternativas para evaluar el desempeño de los algoritmos de agrupación y clasificación para datos estructurados. En el marco de los algoritmos de agrupación en redes, se pueden encontrar algunas medidas para evaluar la calidad de una partición particular del grafo (véase a Fortunato en [37] para una revisión general). En particular, el concepto de modularidad introducido en [43] juega un papel clave en nuestro enfoque. Sin embargo, las medidas existentes no han sido diseñadas para la evaluación del rendimiento de un algoritmo jerárquico, en el cual el resultado final es un dendograma, y no una sola partición. Existen muchas situaciones reales en las cuales las relaciones entre los objetos vienen dadas de forma imprecisa, especialmente en el campo de las redes sociales (ver, [48]). La representación de relaciones entre objetos por medio de grafos nos ha permitido desarrollar un enfoque interesante que encaja en este tipo de problemas y sin una complejidad computacional excesiva. Un trabajo futuro puede ser el de explorar cómo extender las ideas del algoritmo que hemos presentado en un marco más general de "soft-computing" (véase, por ejemplo, [76] para una discusión general del concepto de soft-computing).

5.2. Segmentación jerárquica de imágenes

Uno de los principales problemas cuando se trabaja algoritmos de segmentación de imágenes usando teorías que implican umbrales, es elegir un conjunto de umbrales apropiados para el proceso de segmentación . Si queremos dar el umbral de forma explícita, necesitamos manipular los valores de los píxeles en un espacio de color, y las distancias entre ellos en un espacio de medida . Sin embargo, en el algoritmo D&L que

hemos propuesto es más importante el número de aristas de corte (aristas con valores grandes de disimilaridad) incluidos en una etapa dada de la segmentación, que el valor del umbral α_i; es decir, el algoritmo D&L permite usar un incremento ω en lugar de un conjunto de umbrales. Debido a esto, lo único que se necesita en la segmentación jerárquica que se propone es el número de pasos (el valor K), y el número de aristas de corte ω a ser incorporados en cada paso.

En el procesamiento de las imágenes de las Figuras 4.8 y 4.11, el tiempo CPU fue de no más de 1 minuto para cada imagen en la segmentación jerárquica . El algoritmo se ejecutó en un Intel Core i5 de 1,7 GHz con 3 GB de RAM. Los cinco umbrales se han elegido de forma automática por propio algoritmo con el fin de incluir un tamaño fijo de bordes de corte ($100\,t^2$ aristas de corte se incluyen en cada paso, para $t = 1, \ldots, 5$).

A continuación se muestra una forma de representar de manera condensada una segmentación jerárquica de una imagen con el fin de obtener información adicional en relación con los bordes de los grupos. La técnica consiste en utilizar la secuencia de "salidas" que ofrece el algoritmo D&L y darle un "peso" a los bordes, dependiendo del valor de disimilaridad en el que se creó dicho borde. Así, por ejemplo, el borde de un grupo que se hubiese formado al principio de la segmentación (disimilitud alta), tendrá un borde más intenso, y el borde de un grupo creado al final de la segmentación (disimilitud baja) tendrá un borde tenue. La Figura 5.1 muestra cómo se puede representar la información de la jerarquización.

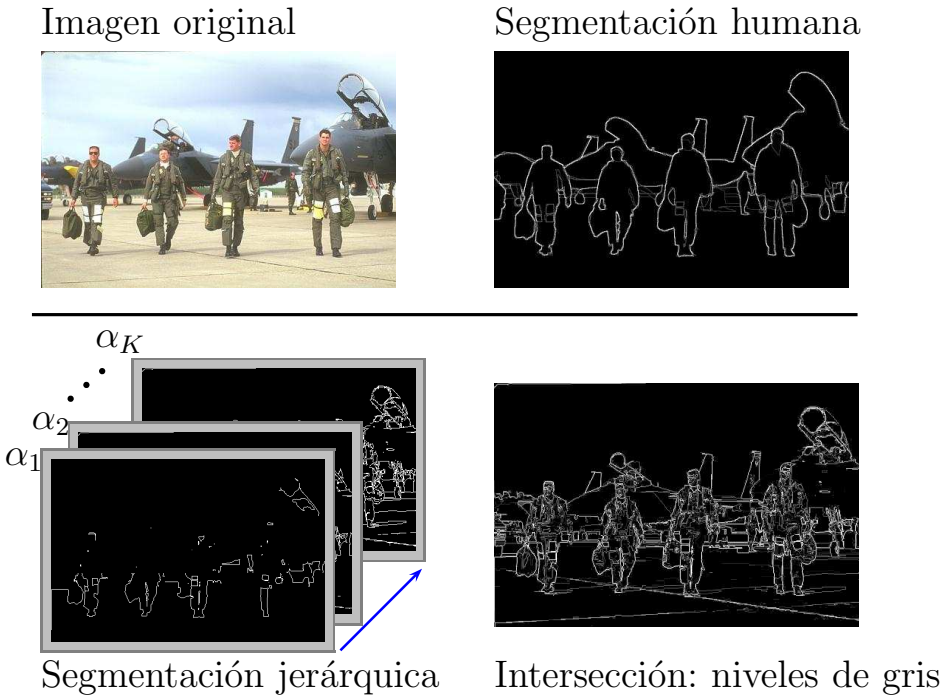


Figura 5.1: Intersección en grises

Como un trabajo futuro se espera formalizar esta información de tonalidades (no necesariamente en tonos de grises) de forma tal que, dependiendo de los objetivos del problema, el investigador pueda identificar de manera más precisa las regiones adyacentes con mayor (o menor) disimilitud en una única "salida" del algoritmo.

Por último, el algoritmo D&L que proponemos puede ser ejecutado usando redes gruesas, lo que reduce sustancialmente el tiempo de cálculo. La teoría de funciones de agregación se puede utilizar para este propósito, es decir, relacionar conjuntos de píxeles

a través de alguna función de agregación, por lo que se pueden obtener resultados más rápidamente y preservar una buena agrupación. Un trabajo futuro de investigación será el estudio de diferentes funciones de agregación en este tipo de redes.

5.3. Imágenes Astronómicas

El algoritmo D&L aplicado a imágenes astronómicas ha mostrado ser una herramienta relevante pues se puede calibrar para el uso de este tipo de imágenes para encontrar estructuras, que no necesariamente sean las más intensas. Además, el uso de D&L en las experiencias computacionales dos ejemplos mostrados en las Figuras 4.20 y 4.21, se lograron obtener con un CPU time menor de un minuto (usando un Intel Core i5 de 1.7GHz con 3GB de RAM). Dicho tiempo se puede reducir aun más (a menos de un segundo) utilizando un cubrimiento más grueso o un α_τ mayor.

Esta observación es importante, pues en el proyecto WSO-UV se espera trabajar con volúmenes de imágenes muy grandes, lo cual vuelve imprescindible el uso de herramientas de ejecución no supervisadas, y que además sean de rápida ejecución.

Finalmente, un trabajo futuro en esta línea es automatizar el cálculo de los "umbrales adecuados" para obtener las regiones débiles (por ejemplo, con un muestreo en la imagen, y luego utilizar series de tiempo o análisis de Fourier, para modelar la secuencia de umbrales). Esto es debe a que dichos umbrales varían entre una imagen y otra, ya sea por las diferencias de exposición de la imágenes o por las diferencias propias entre ellas.

5.4. Segmentación jerárquica borrosa

Desde un punto de vista clásico, el proceso de segmentación de una imagen es un caso particular de un problema de cluster, en el cual, la única información disponible es la de los atributos contenidas en cada pixel. En [49], se presenta y analiza un desarrollo deficiente para la segmentación, que utiliza únicamente la información contenida en cada pixel (datos de atributos). Así, es necesario considerar también la estructura asociada a los datos o pixeles. En esta sección consideraremos relaciones difusas entre los pixeles.

Sea V el conjunto de pixeles y k el número de características o variables asociadas a cada pixel $p \in V$. Nótese que, en el caso de una imagen digital, k será igual a tres, si estamos en el formato RGB, correspondiente con el espectro visible de la imagen. Para imágenes complejas, como por ejemplo, las imágenes de detección remota; los valores de k son superiores. Así, podremos identificar a la i-ésima caracteristica o variable de del item p con la notación en subíndice p_i para $i \in \{1, 2, \ldots, k\}$.

En [49], se definió un grafo valuado para representar una imagen basada en una relación entre pixeles y una función de distancia para representar la disimilaridad entre pixeles. Por otro lado, dado que la distancia entre dos elementos suele ser tema de discusión, y para permitir más flexibilidad que la ofrecida por los procedimientos nítidos (crisp); se propone introducir incertidumbre difusa y distancias difusas (o borrosas) para representar esta disimilaridad ente pixeles (ver [11, 12, 13, 14, 15, 16, 66, 119]). Por lo tanto, si

$$d : V \times V \longrightarrow [\widetilde{0, \infty}) \tag{5.1}$$

representa la relación entre dos propiedades medidas de los items, $\widetilde{[0,\infty)}$ representa el conjunto de números borrosos con dominio en $\mathbb{R}^+$. Luego, si

$$\widetilde{d_{pp'}} = d(p,p') \tag{5.2}$$

representa la relación entre los pixeles p y p', su función de membresía (o pertenencia) será denotada por el mapeo:

$$\mu_{pp'} : \mathbb{R}^+ \longrightarrow [0,1] \tag{5.3}$$

y la matriz de distancias difusas asociadas se denotará por

$$\widetilde{D} = \left\{ \widetilde{d_{pp'}} \;\mid\; (p,p') \in V \times V \right\} \tag{5.4}$$

Definición 5.1: *Dada una función de distancia difusa d, el 'grafo difuso de pixeles' será definido como el par $\widetilde{G(V)} = (V, \widetilde{E})$, donde el valor 'difuso' para una arista asociada a dos items adyacentes p y p' es la distancia difusa entre ellos, es decir:*

$$\widetilde{E_{p,p'}} = \widetilde{d_{p,p'}}. \tag{5.5}$$

Observación: Dados dos pixeles adyacentes p y p', se denotará por $\widetilde{d_{p,p'}}$ como d_e para todo $e \in \widetilde{E}$.

Algoritmo de segmentación borroso

Nuestro propósito acá es agrupar los V pixeles a través de un algoritmo iterativo de coloreado binario de un grafo difuso $\widetilde{G(V)}$. Cualquier coloreado binario de un grafo (ver [83]), induce una partición del conjunto V, siendo cada clase asociada a un color: $V_C(k) = \left\{ i \in V \mid C(i) = k,\ k \in \{1,2,\ldots,c-1\} \right\}$.

5.4.1. Proceso de coloreado binario

Dado un pixel, $p \in V$, coloreado de forma binaria, $\big(\mathrm{col}(i) \in \{0,1\}\big)$; el color de cualquier pixel adyacente $p' \in V$ dependerá del valor 'difuso' $\widetilde{d_{pp'}}$ cuando es comparado con un umbral α dado:

$$\mathrm{col}(p') = \begin{cases} col(p) & \text{si } \widetilde{d_{pp'}} = d_e \leq \alpha \\ 1 - \mathrm{col}(p) & \text{si } \widetilde{d_{pp'}} = d_e > \alpha \end{cases} \tag{5.6}$$

para todo $e \in \widetilde{E}$. Nótese que en la ecuación (5.6) es necesario conocer si el valor d_e es mayor o menor que el umbrar fijado α. El problema de ordenación de los números difusos ha sido estudiado por varios autores (ver [27, 34, 103]). Un procedimiento interesante sería transformar los números difusos en números reales por medio de funciones de rango. Sea F una función de rango en el sentido dado en [27]; diremos entonces que un conjunto difuso $\widetilde{A}$ con función de membresía μ_A y dominio en $\mathbb{R}^+$ es considerado como un número difuso si y sólo si (ver [103]):

- $A_\alpha = \left\{ x \in \mathbb{R}^+ \mid \mu_A \geq \alpha \right\}$ es un conjunto convexo, denotado por $\left[\ \underline{A_\alpha}\ ,\ \overline{A_\alpha}\ \right]$.
- μ_a es una función semi-continua superiormente.

- $\widetilde{A}$ es normal, es decir; existe un $x \in \mathbb{R}^+$ tal que $\mu_A(x) = 1$.
- $\text{supp}(A) = \{x \mid \mu_A(x) > 0\}$ es un conjunto acotado en $\mathbb{R}^+$.

Definición 5.2: *Sea $\mathcal{N}$ un conjunto de números borrosos, y sean $a, b \in \mathcal{N}$. Entonces*

$$a \widetilde{\geq} b \longleftrightarrow F(a) \geq F(b).$$

Con lo anterior expuesto, el problema de ordenación de los números borrosos ha sido resuelto; así, estamos en condiciones de continuar con el proceso jerárquico.

5.4.2. Modelo borroso jerarquizado

Dada una red N y una familia de valores de umbrales

$$\{\alpha_1 > \alpha_2 > \cdots > \alpha_K\} \tag{5.7}$$

donde $\alpha_1 = \text{mín}_{e \in E}\{d_e\}$ y $\alpha_K = \text{máx}_{e \in E}\{d_e\}$. Definimos la siguiente familia de grafos parciales:

$$\left\{ \widetilde{G(V^i)} = (V, \widetilde{E^i}) \quad | \quad i \in \{1, 2, \ldots, K\} \right\} \tag{5.8}$$

donde $\widetilde{E^i} = \{e \in \widetilde{E} \mid d_e \geq \alpha_i\}$. Construyamos la familia de bosques soporte $F^i = (P, T^i)$ con los siguientes dos pasos:

1. Sea $F^i = (V, T^i)$ el mínimo bosque soporte de los grafos parciales $\widetilde{G(V^{i+1})} = (V, \widetilde{E^i} - \widetilde{E^{i+1}})$. Siguiendo un algoritmo similar al de Kruskal, la construcción de F^i es basada en un ordenación de las aristas cuyos pesos pertenecen al intervalo $(\alpha_i, \alpha_{i+1}]$ en orden decreciente.

2. Si F^i así construido es ya un bosque soporte de $\widetilde{G(V^{i+1})}$, terminar; en otro caso, entonces las aristas por fuera de $\widetilde{E^i}$ (aristas menores que α_i) serán ordenadas en orden creciente, y luego serán añadidas iterativamente a T^i (si no forman ciclos), y finalizamos el proceso cuando F^i sea el bosque soporte.

Observaciones.

- La formación de subgrupos homogéneos para un valor α determinado, tiene en cuenta la estructura de los subgrupos formados por el α anterior.
- La condición anterior producirá, en segmentación de imágenes, una secuencia de imágenes donde se aprecia los niveles de segmentación.
- La implementación computacional para el proceso borroso se hace de forma semejante a la del proceso nítido, con la diferencia que se modifica la forma de cuantificar los pesos de los enlaces entre pixeles (en el grafo asociado).

5.4.3. La clases borrosas borde y no borde y el grafo borroso consistente

De todo lo dicho en esta sección, se puede observar que el algoritmo que se presenta obtiene, para cada α, una partición/segmentación de la red. De manera que las aristas pueden dividirse entre aquellas que conectan nodos en un mismo grupo y aquéllas que conectan diferentes grupos homogéneos. En base a esto es posible construir las siguientes funciones de pertenencia μ_B, $\mu_{NB} : E \longrightarrow [0,1]$ que corresponderían a las clases *borde* y *no borde* en el proceso de segmentación:

$$\mu_B(e) = \ Max\big\{\alpha \in [0,1] \ \mid \ \mathrm{col}_\alpha(i) \neq \mathrm{col}_\alpha(j) \ \big\},$$

donde $e = \{i, j\} \in E$. Además,

$$\mu_{NB}(e) = n(\mu_B(e)),$$

siendo n una función de negación.

Obsérvese también que desde la clase borrosa $\{\mu_B\}$ es posible construir un grafo borroso consistente (es decir un grafo sin inconsistencias en el proceso de coloración binario). Sea $\widetilde{G} = (V, \mu_B)$ el grafo borroso dado por la función de pertenencia de la clase *borde*. Teniendo en cuenta cómo se ha construido esta clase, se puede garantizar que $\forall \alpha \in [0,1]$ no es posible encontrar una cadena $\{i_1, \ldots, i_k\}$ en $G = (V, E)$ tal que $d_{i_l, i_{l+1}} \leq \alpha$ y $d_{i_1, i_k} \geq \alpha$. Por ejemplo, sea $G = (V = \{1,2,3,4\}, E = \{d_{1,2} = 0{,}8, d_{1,3} = 0{,}6, d_{2,4} = 0{,}7, d_{3,4} = 0{,}1\})$ un grafo valorado con inconsistencias (por ejemplo para $\alpha = 0{,}75$). Después del proceso de segmentación jerárquico el grafo borroso consistente dado por la clase borde es: $\widetilde{G} = (V = \{1,2,3,4\}, E = \{\mu_B(1,2) = 0{,}8, \mu_B(1,3) = 0{,}6, \mu_B(2,4) = 0{,}8, \mu_B(3,4) = 0{,}1\})$ que no tiene ciclos inconsistentes para ningún valor de α.

Apéndice A

Algoritmos

Contenido

En este apéndice describiremos, paso a paso, la manera de construir e implementar el procedimiento de segmentación jerarquizada, propuesto en los capítulos anteriores, para el caso de imágenes. Inicialmente se ilustrará de una forma general, los pasos del algoritmo asociado al proceso (Sección A.2), y en las secciones posteriores los detalles técnicos más relevantes en el proceso.

A.1. Algoritmo basado en búsqueda matricial

A continuación mostraremos los pasos realizados para obtener una segmentación jerarquizada de una imagen a través de un algoritmo que manipula la información de los nodos desde una matriz, de dimensión $N \times 4$, donde N es el número de nodos, y 4 es la cantidad de vecinos por nodo. Note que este algoritmo tiene como base un tipo de vecindad en particular (como lo muestra la Figura A.1). A continuación daremos una descripción de este algoritmo, cuyo código Matlab está se encuentra en la Sección A.3, y el nombre del archivo principal es `hseg4.m` (ver Subsección A.3.1).

A.1.1. Topología de la red

Para el proceso de segmentación que queremos desarrollar usaremos una topología basada en la siguiente relación de adyacencia: dos pixeles $p = (i, j)$ y $p' = (i', j')$ son adyacentes cuando ellos comparten una coordenada, siendo la otra contigua. De esta forma podemos definir el conjunto de aristas E de un grafo planar $G = (P, E)$ como:

$$E = \left\{ \big((i,j),(i',j')\big) \in P \;\middle|\; \{i = i' \text{ y } |j - j'| = 1\} \text{ ó } \{j = j' \text{ y } |i - i'| = 1\} \right\} \quad \text{(A.1)}$$

Para una imagen digital de dimensiones $r \times s$, la Figura A.1 muestra los vecinos que se relacionan con el pixel (i, j) de la imagen digital. Nótese que, dada una enumeración horizontal secuencial de los pixeles, entonces, al pixel P_k en la posición (i, j) le corresponderá como subíndice $k = (i-1)s + j$, para $i = 1, 2, \ldots, r \quad \text{y} \quad j = 1, 2, \ldots, s$.

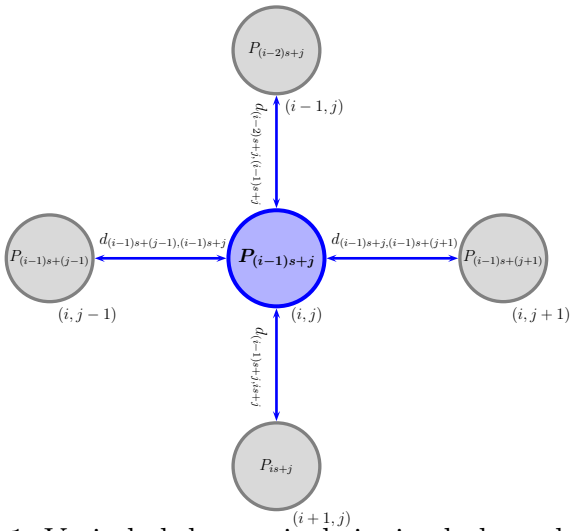


Figura A.1: Vecindad de un pixel siguiendo la regla (A.1).

La representación mostrada en (A.1) induce un arreglo matricial para los pixeles con la siguiente estructura:

$$G = \begin{bmatrix} P_1 & P_2 & P_3 & \cdots & P_s \\ P_{s+1} & P_{s+2} & P_{s+3} & \cdots & P_{2s} \\ P_{2s+1} & P_{2s+2} & P_{2s+3} & \cdots & P_{3s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ P_{(r-1)s+1} & P_{(r-1)s+2} & P_{(r-1)s+3} & \cdots & P_{rs} \end{bmatrix}$$

donde el $P_{(i-1)s+j}$ hace referencia al pixel en la posición (i, j). La Figura A.2 muestra el esquema de las estructura propuesta en (A.1) para una imagen.

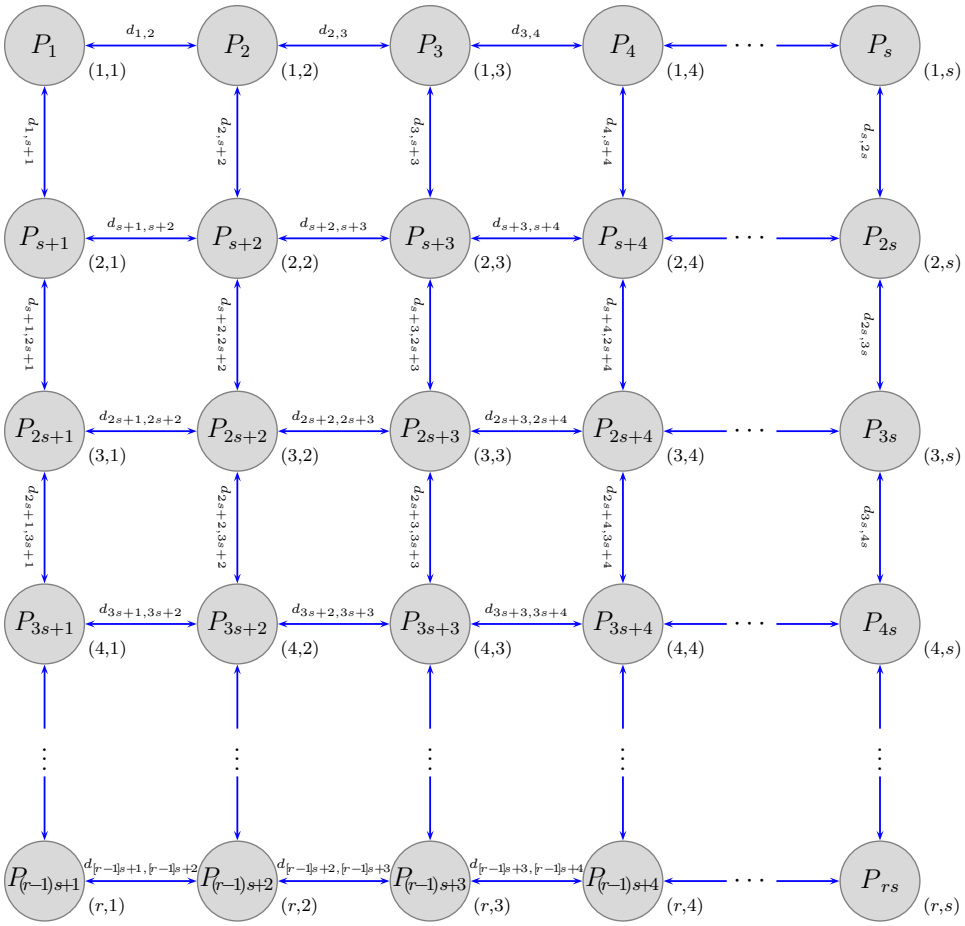


Figura A.2: Topolología de la red (imagen como un grafo)

A.1.2. Concentración de la información. Reducción

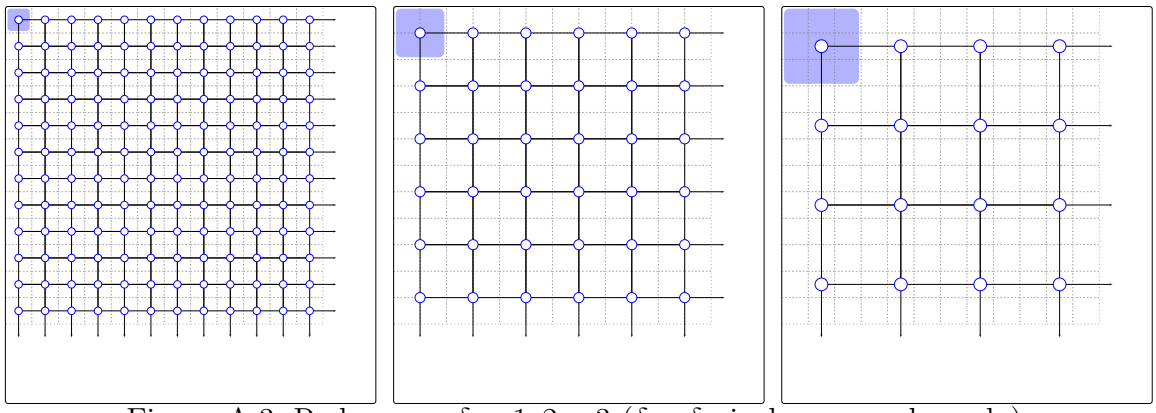


Figura A.3: Redes para $\delta = 1, 2$ y 3 ($\delta \times \delta$ pixeles por cada nodo).

La información que va a representar cada nodo será entonces la de un pixel, o la de un conjunto (de pixeles) $\Omega = \delta^2$. Esto último es lo que se denomina "concentrar" información, a través de algún operador de agregación $\mathcal{A}$, como se muestra en la Figura A.4.

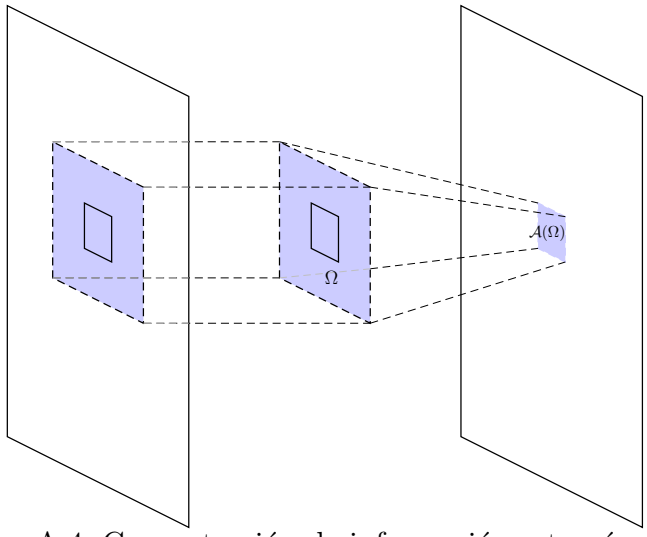


Figura A.4: Concentración de información a través de $\mathcal{A}$.

A.1.3. Distancias

El cálculo de la medida de disimilitud en imágenes se realiza justo después de tener definido un conjunto de nodos; en donde cada nodo representará un pixel o un grupo de pixeles ($\mathcal{A}(\Omega)$). El algoritmo `hseg4.m` trabaja con dos tipos de imágenes: de intensidad y RGB. Para el caso de imágenes de intensidad, lo único que se debe hacer antes de crear las distancias para la red mostrada en la Figura A.2 es garantizar que se estén en formato `double` para poder realizar todas las operaciones con los valores de los nodos.

Finalmente, se considera una distancia entre pixeles para calcular los valores de las aristas. Para el caso de imágenes de intensidad, la distancia habitual es la Euclídea

en $\mathbb{R}$, y en el caso de imágenes a color, la medida CIE76. Note que, para una imagen de dimensión $r \times s$, el número de aristas para generar la red de la Figura A.1 es $(r-1)s + (s-1)r$. Así por ejemplo, para una imagen de $1000 \times 1000 = 10^6$ pixeles, se deben generar $1998000 \approx 2 \times 10^6$ aristas. Debido a la gran cantidad de conexiones, se optimizó el proceso usando operaciones matriciales en lugar de operaciones entre cada par de nodos.

Umbrales

Luego de tener constituida una red para la imagen, lo siguiente (y primordial) en el proceso de segmentación jerárquica, es incluir (o generar) un conjunto de umbrales los cuales van a permitir hacer la jerarquización y clasificar las aristas en aristas de "disimilitud" y aristas de "afinidad". El algoritmo permite `hseg4.m` permite ingresar por el usuario un conjunto de umbrales

$$\{\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_K\}, \tag{A.2}$$

o, simplemente, el valor K: profundidad de la jerarquización o número de segmentaciones. El inconveniente de que un usuario deba ingresar un conjunto de umbrales, como el mostrado en (A.2), es que necesariamente estos valores de α se mueven en el espacio final de medida de los pixeles, lo cual hace difícil la escogencia de un conjunto de umbrales adecuado.

Submatrices

Una forma bastante eficiente de aplicar nuestro algoritmo de segmentación jerárquica, es usando submatrices conectadas entre si por las respectivas aristas "frontera", que permitirán que la información sobre la generación de grupos fluya entre submatrices adyacentes.

A.2. Algoritmo con Búsqueda por Punteros

Para implementar el algoritmo de segmentación al caso de imágenes, es fundamental conocer el tipo de imagen con la que se va a trabajar, en particular el formato y el tamaño; además de los objetivos específicos que se busca con la segmentación. Así por ejemplo, para el caso de imágenes astronómicas que por lo general suelen ser de un tamaño de 3840×3840, un objetivo puede ser analizar un conjunto grande de dichas imágenes (cien mil o doscientos mil imágenes) a través de segmentación. El conocimiento de este objetivo específico permite entonces tomar decisiones como el tamaño de mallado que se va a asociar a las imágenes, ésto para mejorar el tiempo de ejecución (pues por defecto, el tamaño es igual al de la imagen).

A.2.1. Generación de una red

Una manera de construir una red asocial a una imagen dada I, con $r \times s$ pixeles, es recorriendo ordenadamente la matriz ($i = 1 \to r$, and $j = 1 \to s$) e ir creando los enlaces dependiendo de la topología y la medida escogida (como lo muestra la Figura A.5).

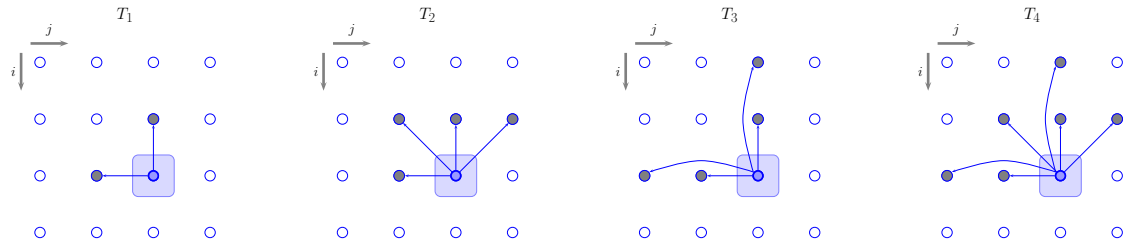


Figura A.5: Network construction when a particular T_i is used.

A.2.2. Almacenamiento de matrices dispersas

Uno de los principales problemas asociados al problema de segmentación de una red es el almacenamiento de las matrices asociadas a dicha red, muchas de las cuales son las llamadas matrices *dispersas* (matrices con "muchos" ceros). Es deseable poder almacenar estas matrices en la memoria del ordenador minimizando la cantidad de espacio utilizado para dicho propósito. Nuestro propósito es extender la forma de almacenar este tipo de matrices, usados en métodos directos e iterativos, para la resolución de sistemas lineales (dependiendo del tipo de matriz -simétrica o no-). Esta metodología es ampliamente usada matemáticas aplicadas y computación; por ejemplo, en la resolución de problemas de contorno en ecuaciones diferenciales parciales a través del método de los elementos finitos (ver).

Caso simétrico

Las ideas que desarrollaremos a continuación las ilustraremos con el ejemplo de la matriz simétrica ($a_{ij} = a_{ji}$):

$$A = \begin{pmatrix} a_{11} & a_{12} & 0 & a_{14} & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & 0 & a_{43} & a_{44} & a_{45} \\ 0 & 0 & a_{53} & a_{54} & a_{55} \end{pmatrix} \tag{A.3}$$

Método directo

La matriz se almacena en dos vectores $M1$ y $M2$. El vector $M1$ funciona a modo de *puntero* y en el vector $M2$ se guardan todos los elementos que están entre el primer elemento no nulo de cada fila y el de la diagonal. El vector $M1$ tiene tantas coordenadas como el número de nodos más 1 (en el ejemplo, 6). En él se almacenan, acumulativamente, la cantidad de elementos que se deben guardar de cada fila (es decir, los que hay entre el primer elemento no nulo y el de la diagonal). Siempre se toma $M1(1) = 0$. Así, en el ejemplo anterior

$$M1 = (0, 1, 1+2, 3+2, 5+4, 9+3) = (0, 1, 3, 5, 9, 12).$$

En cuanto al vector $M2$, su número de coordenadas será $M1$ (número de nodos + 1), es decir, el valor de la última coordenada de $M1$ (en nuestro ejemplo, 12). Se guardan en él, por orden de filas, los valores de los elementos correspondientes. En (A.3),

$$M2 = \big(\underbrace{a_{11}}_{\text{fila 1}} , \underbrace{a_{21}, a_{22}}_{\text{fila 2}} , \underbrace{a_{32}, a_{33}}_{\text{fila 3}} , \underbrace{a_{41}, 0, a_{43}, a_{44}}_{\text{fila 4}} , \underbrace{a_{53}, a_{54}, a_{55}}_{\text{fila 5}} \big).$$

De esta forma, los elementos que se guardan de la fila i están almacenados en las posiciones

$$M1(i)+1, M1(i)+2, \ldots, M1(i+1)$$

del vector $M2$; por tanto, de dicha fila se guardan $M1(i+1)-M1(i)$ elementos. Además, es sencillo recuperar la matriz A: si a_{ij} es un elemento de los que se almacenan, entonces

$$a_{ij} = M2\big(\, M1(i+1) - i + j \,\big),$$

es decir, para cada $i = 1, 2, ..., N$

$$a_{ij} = \begin{cases} M2\big(\, M1(i+1) - i + j \,\big), & j = i - \big(M1(i+1) - M1(i)\big) + 1, \ldots, i \\ 0, & j = 1, 2, \ldots, \big(M1(i+1) - M1(i)\big). \end{cases}$$

Esta forma de almacenar la matriz se suele denominar de tipo perfil. Como es sabido, los métodos directos no conservan los ceros de la matriz (se produce el fenómeno del fill–in o rellenado). Es fácil demostrar, sin embargo, que dichos métodos efectivamente preservan lo que se denomina el perfil de la matriz, es decir, mantienen intactos los ceros que se encuentran antes del primer elemento no nulo de cada fila. En el ejemplo anterior, el perfil de la matriz está formado por los elementos

$$\begin{pmatrix} \times & & & & \\ \times & \times & & & \\ & \times & \times & & \\ \times & \times & \times & \times & \\ & & \times & \times & \times \end{pmatrix}$$

Ejemplo A.1.
La matriz A cuyo perfil se ha almacenado en los vectores $M1 = (0, 1, 3, 4, 8)$ y $M2 = (9, 8, 7, 5, 1, 0, 1, 2)$ es

$$\begin{pmatrix} 9 & 8 & 0 & 1 \\ 8 & 7 & 0 & 0 \\ 0 & 0 & 5 & 1 \\ 1 & 0 & 1 & 2 \end{pmatrix}$$

Método iterativo

Si se va a emplear un método iterativo para resolver el sistema, la matriz (su parte triangular inferior) se almacenará utilizando tres vectores $L1$, $L2$ y $L3$ donde el primero actuará como puntero, en el segundo se guardarán las columnas de los elementos no nulos y en el tercero sus valores. Nótese que, por tanto, sólo se almacenan los elementos no nulos.

El vector $L1$ tiene tantas coordenadas como el número de nodos más 1 (en el ejemplo, 6). En él se guardan, acumulativamente, el número de elementos no nulos que hay en cada fila hasta la diagonal. Se toma siempre $L1(1) = 0$.

El vector $L2$ tiene $L1$ (número de nodos + 1) coordenadas y en él se almacenan, consecutivamente, las columnas en que se encuentran los elementos no nulos de cada fila hasta la diagonal.

El vector $L3$ tiene el mismo número de coordenadas que $L2$ y guarda los valores de los elementos correspondientes. Para el ejemplo con el que estamos trabajando se tendría:

$$\begin{aligned} L1 &= (0, 1, 1+2, 3+2, 5+3, 8+3) = (0, 1, 3, 5, 8, 11) \\ L2 &= \Big(\boxed{1}, \boxed{1,2}, \boxed{2,3}, \boxed{1,3,4}, \boxed{3,4,5} \Big) \\ L3 &= \big(\underbrace{a_{11}}_{\text{fila 1}}, \underbrace{a_{21}, a_{22}}_{\text{fila 2}}, \underbrace{a_{32}, a_{33}}_{\text{fila 3}}, \underbrace{a_{41}, a_{43}, a_{44}}_{\text{fila 4}}, \underbrace{a_{53}, a_{54}, a_{55}}_{\text{fila 5}} \big). \end{aligned}$$

En este caso, los $L1(i+1) - L1(i)$ elementos no nulos de la fila i están almacenados en las posiciones

$$L1(i)+1, L1(i)+2, \ldots, L1(i+1)$$

del vector $L3$, siendo sus columnas:

$$L2\big(L1(i)+1\big), L2\big(L1(i)+2\big), \ldots, L2\big(L1(i+1)\big)$$

Nótese que $i = L2\big(L1(i+1)\big)$ y $a_{ii} = L3\big(L1(i+1)\big)$.

Cuando se utiliza un método iterativo en la resolución de un sistema lineal se necesita conocer, no tanto la matriz, como el resultado de multiplicar ésta por vectores arbitrarios. Veremos, mas adelante, que es muy sencillo implementar la multiplicación de una matriz por un vector cuando se tiene la matriz almacenada en la forma anteriormente descrita. Es por ello por lo que no es necesario preocuparse de cómo recuperar la matriz a partir de los vectores $L1$, $L2$ y $L3$ (aunque puede hacerse de manera sencilla).

Ejemplo A.2.
La matriz A del Ejemplo A.1 se almacenaría en este caso mediante los vectores:

$$L1 = (0, 1, 3, 4, 7), \quad L2 = (1, 1, 2, 3, 1, 3, 4) \quad \text{y} \quad L3 = (9, 8, 7, 5, 1, 1, 2).$$

Caso no simétrico

Método directo

También aquí se utilizará un almacenamiento de tipo perfil: de cada fila únicamente se guardarán los elementos comprendidos entre el primero y el último no nulos. En este caso, necesitaremos tres vectores $M1$, $M12$ y $M2$. El vector $M1$ desempeñará un papel análogo al del caso simétrico; tiene tantas coordenadas como el número de nodos más 1 y en él se almacenan, acumulativamente, la cantidad de elementos que se deben guardar de cada fila. También ahora se toma $M1(1) = 0$.

El vector $M12$ tiene tantas coordenadas como el número de nodos y en cada componente se guarda la columna del primer elemento no nulo de la fila correspondiente.

Finalmente, en el vector $M2$ se guardan, por orden de filas, los valores de los elementos correspondientes.

Para una matriz de la forma:

$$A = \begin{pmatrix} a_{11} & a_{12} & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 \\ 0 & 0 & a_{33} & a_{34} & a_{35} \\ a_{41} & 0 & a_{43} & a_{44} & a_{45} \\ 0 & a_{52} & a_{53} & a_{54} & a_{55} \end{pmatrix} \tag{A.4}$$

se obtiene:

$$\begin{aligned} M1 &= (0,2,5,8,13,17) \\ M12 &= (1,1,3,1,2) \\ M2 &= \big(\underbrace{a_{11},a_{12}}_{\text{fila 1}}\,,\,\underbrace{a_{21},a_{22},a_{23}}_{\text{fila 2}}\,,\,\underbrace{a_{33},a_{34},a_{35}}_{\text{fila 3}}\,,\,\underbrace{a_{41},0,a_{43},a_{44},a_{45}}_{\text{fila 4}}\,,\,\underbrace{a_{52},a_{53},a_{54},a_{55}}_{\text{fila 5}}\big). \end{aligned}$$

La matriz así almacenada puede recuperarse de la siguiente forma:

$$a_{ij}=\begin{cases} M2\big(M1(i)+j-M12(i)+1\big), & j=M12(i),\ldots,M12(i)+\big(M1(i+1)-M1(i)\big)-1 \\ 0, & j\le M12(i)-1 \text{ y } j\ge M12(i)+\big(M1(i+1)-M1(i)\big) \end{cases}$$

para $i=1,2,\ldots,N$.

Método iterativo

Como en el caso simétrico, se almacenan sólo los elementos no nulos. En $L1$ se guardan, acumulativamente, el número de elementos no nulos de cada fila, en $L2$ sus columnas (colocando, para cada fila, en último lugar la columna del elemento diagonal) y en $L3$ los coeficientes correspondientes en ese orden. Así, para la matriz anterior, se tendría:

$$\begin{aligned} L1 &= (0,2,5,8,12,16) \\ L2 &= \big(\boxed{2,1}\,,\,\boxed{1,3,2}\,,\,\boxed{4,5,3}\,,\,\boxed{1,3,5,4}\,,\,\boxed{2,3,4,5}\big) \\ L3 &= \big(\underbrace{a_{22},a_{11}}_{\text{fila 1}}\,,\,\underbrace{a_{21},a_{23},a_{22}}_{\text{fila 2}}\,,\,\underbrace{a_{34},a_{35},a_{33}}_{\text{fila 3}}\,,\,\underbrace{a_{41},a_{43},a_{45},a_{44}}_{\text{fila 4}}\,,\,\underbrace{a_{52},a_{53},a_{54},a_{55}}_{\text{fila 5}}\big). \end{aligned}$$

A.2.3. Técnica de "perfil" en la generación del árbol soporte

Uno de los procedimientos que más coste computacional tiene a la hora de utilizar el algoritmo D&L para es la generación del árbol soporte asociado dicha técnica. Esto se debe, principalmente, al hecho que el árbol se construye de manera "aglomerativa", es decir; se van incorporando uno a uno las aristas que lo forman, y eso implica que en algunos momentos una arista puede estar uniendo "dos ramas" ya formadas, y así esas dos componentes conexas pasarían a ser una sola componente conexa. Lo anterior muestra el espíritu del proceso D&L en el sentido que ir incluyendo aristas equivale a estar uniendo grupos (o ramas del árbol). En general, la dinámica de unir grupos implica darles una misma asignación de "nombre". Para resolver este problema, se hace entonces necesario usar técnicas que me permita identificar los grupos, unirlos (es decir, asignarles el mismo nombre) y continuar el proceso de construcción con ésta actualización; todo esto, de forma eficiente.

Cuando se solucionan numéricamente problemas de contorno en ecuaciones diferenciales parciales a través del método de los "elementos finitos", también se tiene un problema de identificación y almacenamiento de los "elementos" del dominio (y que aumentan significativamente cuando se hace un refinamiento del dominio). Sin embargo, la forma de almacenar matrices dispersas como se ha explicado anteriormente a través de técnicas de "perfil", permite un uso de menos memoria para almacenar los datos (pues se hace un almacenamiento vectorial) y también permite identificar los "elementos" del dominio.

A continuación se dará una explicación breve el uso de esta técnica de "perfil" para construir el árbol soporte para el algoritmo D&L. Inicialmente, se tiene un vector de grupos Gr (de dimensión k) con los cuales se hará el proceso aglomerativo del formación del árbol; además, supongamos que se tienen una matriz de sumas acumuladas S (de dimensión $k+1$), que va sumando el la cantidad de grupos que forman un nuevo grupo; y sea matriz auxiliar de grupos M_0 (inicialmente, $M0 = Gr$).

Ahora bien, si $k_a = Gr(a)$ y $k_b = Gr(b)$ son dos grupos distintos y se van a juntar, así, $n_a = S(k_a + 1) - S(k_a)$ y $n_b = S(k_b + 1) - S(k_b)$ van a denotar el número de elementos que actualmente conforman los grupos k_a y k_b, respectivamente.

Consideremos entonces las dos posibilidades siguientes:
Si $k_a < k_b$: en este caso, movemos el grupo k_b hacia el grupo k_a. Para lograrlo, el vector $M0$ se divide las siguientes cinco partes:

$$\begin{aligned} M0 &= \Big[\underbrace{1 : k_a - 1}_{M0_1}, \underbrace{k_a : k_a + n_a - 1}_{M0_{k_a}}, \underbrace{k_a + n_a : k_b - 1}_{M0_2}, \underbrace{k_b : k_b + n_b - 1}_{M0_{k_b}}, \underbrace{k_b + n_b : \text{end}}_{M0_3} \Big] \\ &= \big[\, M0_1 \ , \ M0_{k_a} \ , \ M0_2 \ , \ M0_{k_b} \ , \ M0_3 \,\big]. \end{aligned}$$

Así, la nueva estructura de M será

$$M = \big[\, M0_1 \ , \ M0_2 \ , \ M0_{k_b} \ , \ M0_{k_a} \ , \ M0_3 \,\big].$$

A continuación, se actualiza el vector de sumas acumuladas S para continuar el proceso.
Si $k_a \geq k_b$: en este caso, movemos k_a hacia k_b. $M0$ tiene, para este caso, la siguiente forma:

$$\begin{aligned} M0 &= \Big[\underbrace{1 : k_b - 1}_{M0_1}, \underbrace{k_b : k_b + n_b - 1}_{M0_{k_b}}, \underbrace{k_b + n_b : k_a - 1}_{M0_2}, \underbrace{k_a : k_a + n_a - 1}_{M0_{k_a}}, \underbrace{k_a + n_a : \text{end}}_{M0_3} \Big] \\ &= \big[\, M0_1 \ , \ M0_{k_b} \ , \ M0_2 \ , \ M0_{k_a} \ , \ M0_3 \,\big]. \end{aligned}$$

Luego, al mover el grupo k_b hacia k_a, la estructura de M será:

$$M = \big[\, M0_1 \ , \ M0_2 \ , \ M0_{k_a} \ , \ M0_{k_b} \ , \ M0_3 \,\big]$$

Finalmente, se actualiza el vector de sumas acumuladas S, y así continuar con el proceso iterativo de construcción del árbol soporte.

A.3. Algoritmos

A.3.1. hseg4.m

```
% Programa para realizar una segmentación jerárquica de una imagen
% usando el algoritmo D&L. Este proceso combina los métodos de
% reducción y partición de la imagen.

typeim=input('RGB images press "1", intensity images press "0": ');
IMAG=input('input image: ');

if typeim==1  % Para imágenes RGB usamos el espacio Lab
    C=makecform('srgb2lab');
    lab=applycform(IMAG,C);
    labd=lab2double(lab);
    Aredu=labd;
else
    Aredu=IMAG;
end

CONSTAN=input('[reduction, partition i ,partition j]: ');
d=CONSTAN(1); DELTI=CONSTAN(2); DELTJ=CONSTAN(3);
%-------------------------------------------------------------------------
%------------------------- Mallado ---------------------------------------
tic

[r,s,t]=size(Aredu);  S1=floor(s/d);  R1=floor(r/d);
d1=(r/d-R1)*d; d2=(s/d-S1)*d; % Partición del dominio

if d1>0
    R1=R1+1;
end
if d2>0
    S1=S1+1;
end
B=zeros(R1,S1,t);

for K=1:t
 for I=1:R1
    for J=1:S1
        suma=0;
        if I==R1 && d1>0
            da=d1;
        else
            da=d;
        end
        if J==S1 && d2>0
            db=d2;
        else
            db=d;
        end
        for k=1:da
            for l=1:db
                Ik=(I-1)*d+k;  Jl=(J-1)*d+l; suma=suma+Aredu(Ik,Jl,K);
            end
```

```
        end
        B(I,J,K)=suma/(da*db);
    end  % of J
 end  % of I
end  % of K
A=B;
toc
% ------------------------- END mallado-----------------------------------
% -----------------------------------------------------------------------
[r,s,t1]=size(A);   Q=r*s;  % número de nodos
 for t=1:t1
     BB=A(:,:,t)'; PIXdat(t,:)=BB(:);
 end
% ---------------- Posición de submatrices  Ai y  Aj ---------------------
Ai(1,:)=ones(1,s);
for i=2:r
    Ai(i,:)=Ai(i-1,:)+1;
end
Aj(:,1)=ones(r,1);
for j=2:s
    Aj(:,j)=Aj(:,j-1)+1;
end
AT=(Ai-1).*s+Aj;  %-- AT identifica cada nodo con un número entero -----
% -----------------------------------------------------------------------
ATT=AT'; PIXgru1=-ATT(:)'; Aii=Ai'; Ajj=Aj'; PIXi=Aii(:)';
PIXj=Ajj(:)'; PIXcol1=-ones(1,Q); PIXvis1=zeros(1,Q);
PIXgf1=zeros(1,Q); PIXH1=zeros(4,Q); PIXgfA=PIXgf1;
% -----------------------------------------------------------------------
% --Cálculo de la matriz de distancias Euclídeas (topología cuadrada)--
% ------------------------------(Grafo completo)-------------------------
A1=A; A1(:,s,:)=[];  A2=A; A2(:,1,:)=[];
AHE=A1-A2; AH2=AHE.^2;  % aristas horizontales
A3=A; A3(r,:,:)=[]; A4=A; A4(1,:,:)=[];
AVE=A3-A4; AV2=AVE.^2;  % aristas verticales
%-------------------------------Distancias-------------------------------
AHSUM=zeros(r,s-1); AVSUM=zeros(r-1,s);
for t=1:t1
    AHSUM=AHSUM+AH2(:,:,t);  AVSUM=AVSUM+AV2(:,:,t);
end
AH=AHSUM.^(1/2);  % AH(i,j)=dis[A(i,j),A(i,j+1)].
AV=AVSUM.^(1/2);  % AV(i,j)=dis[A(i,j),A(i+1,j)].
%-----------------------------------------------------------------------
%------------------------- Proceso Jerárquico ---------------------------
umb=input('input vector (or number) of thresholding: ');
ATH=AT; ATV=AT; ATH(:,s)=[]; ATV(r,:)=[];
if size(umb,2)==1
    AHH=AH'; D3H=AHH(:); AVV=AV'; D3V=AVV(:);
    Dalp=cat(1,D3H,D3V);  % total distances E
    [a1alp,a2alp]=size(Dalp);
    [M1,IX]=sort(Dalp,'ascend'); Malp=Dalp;
    for j=1:a1alp
        Malp(j)=Dalp(IX(j));
    end
    for i=1:umb   % generación de umbrales
        ALP(i)=Malp(a1alp-20*((umb-i)^2)); % función cuadrática
    end
    ALP(umb+1)=Malp(a1alp);
```

```
else
    ALP=umb;
end
w2=size(ALP,2); %ALP
%-------------------------------------------------------------------------
NH=[1,-s,-1,s];   % topología con 4 vecinos (+)
NH1=[3,4,1,2];    % auxiliar para buscar en PIXH el "dual" del vecino.
A1=zeros(r,s,w2-1); A2L=A1; A2a=A1; A2b=A1; A3=A2; % guardar info
% ------------- Proceso de partición de la matriz-------------------------
PartI=floor(r/DELTI); PartJ=floor(s/DELTJ); extendI=0; extendJ=0;
if PartI<(r/DELTI)
    extendI=1;    % caso final para I
end
if PartJ<(s/DELTJ)
    extendJ=1;    % caso final para J
end
maxI=PartI+extendI; maxJ=PartJ+extendJ;
tic
DHP=[]; DVP=[]; MF=[]; w1F=[]; LLF=[]; DIMIJ=[];% guarda resultado IJ
for I=1:maxI
     auxI=0;
     I1=(I-1)*DELTI+1;
     if I<maxI
         I2=I1+DELTI-1;
     else
        I2=r; auxI=1;
     end
     for J=1:maxJ
        auxJ=0;
        J1=(J-1)*DELTJ+1;
        if J<maxJ
            J2=J1+DELTJ-1;
        else
            J2=s; auxJ=1;
        end
        if J1<s
            DHP=AH(I1:I2,J1:J2-auxJ); % aristas horizontales en  IJ
        end
        if I1<r
            DVP=AV(I1:I2-auxI,J1:J2); % aristas verticales en  IJ
        end
        % ----------------------Bosque soporte IJ ---------------------------
        % DHP y DVP correspondent a las distancia de la parte de  A
        % dada por A(I1:I2,J1:J2) (incluyendo aristas conectoras).
        % ------------------------- Nodos parciales ------------------------
        if J2<s
            ATH=AT(I1:I2,J1:J2);
        else
            ATH=AT(I1:I2,J1:J2-1);
        end
        [rp,sp]=size(DHP);
        if rp>0  % (si existen aristas horizontales)
            AHH=DHP'; ATHH=ATH'; D1H=ATHH(:); D2H=D1H+1; D3H=AHH(:);
            D4H=ones(rp*sp,1); D5H=D4H.*3;
            DH=cat(2,D1H,D2H,D3H,D4H,D5H);
        else
            DH=[];
```

```
166         end
167         % nodos verticales
168         if I2<r
169             ATV=AT(I1:I2,J1:J2);
170         else
171             ATV=AT(I1:I2-1,J1:J2);
172         end
173         [rp1,sp1]=size(ATV);
174         if rp1>0
175             AVV=DVP'; ATVV=ATV'; D1V=ATVV(:); D2V=D1V+s; D3V=AVV(:);
176             D4V=4.*ones(rp1*sp1,1); D5V=2.*ones(rp1*sp1,1);
177             DV=cat(2,D1V,D2V,D3V,D4V,D5V);
178         else
179             DV=[];
180         end
181         %------------------D: matriz de distancia IJ ---------------------
182         D=cat(1,DH,DV); [a1,a2]=size(D); w1=size(D,1);
183         DIMIJ(I,J)=w1;  % guarda  dim(D) (filas)
184         w1F=cat(1,w1F,w1);  % guarda cada dim(D_IJ)
185         %-------------------- M = D  ordenado ------------------------------
186         if w1>0   % if D≠[]
187           [M1,IX]=sort(D(:,3),'ascend');
188           M=D;
189           for j=1:a1
190              M(j,:)=D(IX(j),:);
191           end
192           MF=cat(1,MF,M);  % MF guarda la matriz M_IJ.
193           % - Umbrales en M:  LL guarda las posiciones de los umbrales-
194           %------- Posiciones LL: M(i,3)<ALP(j) ≤ M(i+1,3) -------------
195           i=1;
196           for j=1:w2
197             UMB=ALP(j); ii=1; LL(j)=0;
198              if i<1
199                  i=1;
200              end
201              while i≤w1 && ii==1
202                 if M(i,3)<UMB
203                     i=i+1;
204                 else
205                       i=i-1; ii=0;
206                     if i>0
207                          LL(j)=i;
208                     else
209                        LL(j)=1;
210                     end
211                 end
212              end
213              if LL(j)==0
214                  LL(j)=w1;
215              end
216           end
217             LLF=cat(1,LLF,LL);
218         else
219             LLF=cat(1,LLF,zeros(1,w2));
220         end  % end if w1>0
221         %- LLF guarda pos de LL_IJ  (filas) para cada alpha (columna)--
222         DHP=[]; DVP=[]; LL=[];  % re-iniciar
```

```
223      end
224  end
225  %————————————————————————— Hierarchical process ————————————————————————
226  A1=zeros(r,s,w2-1); %A2L=A1; A2a=A1; A2b=A1; A3=A2;% guarda info final
227  AVERAG=0; % guarda promedio por componentes
228  za=1;   % contador global para los umbrales
229  for ite=w2:-1:2
230    PIXvis=PIXvis1; PIXgru=PIXgru1; PIXcol=PIXcol1; PIXgf=PIXgf1;
231    PIXH=PIXH1; CIndIJ=0; GRUP=0; % asigna etiquetas a los grupos
232    L1=ALP(ite-1); L2=ALP(ite);
233    k=0; kk=0; % kk cuenta componentes conexas
234    %————————————————————Bosque soporte completo ————————————————————————
235    for I=1:maxI
236      for J=1:maxJ
237        %————————————————————————— Bosque soporte IJ ————————————————————————
238       if DIMIJ(I,J)>0
239         CIndIJF=CIndIJ+DIMIJ(I,J); M=MF(CIndIJ+1:CIndIJF,:);
240         CIndIJ=CIndIJF; Li1=LLF((I-1)*maxJ+J,ite-1);
241         Ls2=LLF((I-1)*maxJ+J,ite);
242         i=0; A=[0,0,0,0,0]; ARB=[0,0,0,0,0];
243         for h=Ls2:-1:Li1+1
244             i=i+1; A(i,:)=M(h,:);% filas decreciente (alp_i-1,alp_i]
245         end
246         for h=1:Li1
247             i=i+1; A(i,:)=M(h,:);% filas creciente [alp_0,alp_i-1]
248         end
249         Jpar1=size(A,1); j=1;
250         while j≤Jpar1
251         P1=A(j,1); P2=A(j,2);
252         if PIXgfA(P1)==PIXgfA(P2) % dentro del grupo
253         E1=A(j,4); E2=A(j,5);
254         if PIXvis(P1)==0 && PIXvis(P2)==0  % (nuevo árbol)
255            kk=kk+1;
256            PIXvis(P1)=1;  PIXvis(P2)=1;    % P1 y P2 fueron visitados
257            k=k+1; PIXgru(P1)=k; PIXgru(P2)=k; % conteo de nuevo árbol
258            GRUP(k)=2;    % dos elementos incorporados al grupo k
259            PIXH(E1,P1)=1; PIXH(E2,P2)=1;
260         else
261           if PIXvis(P1)==1 && PIXvis(P2)==0
262              PIXvis(P2)=1;             %  P2 es visitado
263              PIXgru(P2)=PIXgru(P1);    % adicionar P2 al árbol de P1
264              GRUP(PIXgru(P1))=GRUP(PIXgru(P1))+1;% add un elemento
265              PIXH(E1,P1)=1; PIXH(E2,P2)=1;
266           else
267             if PIXvis(P1)==0 && PIXvis(P2)==1
268                PIXvis(P1)=1;           % P1 es vistado
269                PIXgru(P1)=PIXgru(P2);  % adicionar P1 al árbol de P2
270                GRUP(PIXgru(P2))=GRUP(PIXgru(P2))+1;
271                PIXH(E1,P1)=1; PIXH(E2,P2)=1;
272            else    % case when PIXvis(P1)=1=PIXvis(P2)
273                  if PIXgru(P1)≠PIXgru(P2)
274                     kk=kk-1;      % restar un elemento del del árbol
275                   if GRUP(PIXgru(P1))>GRUP(PIXgru(P2))
276                    T=P2;
277                    m=1; n=1;  % m: contador de T
278                   while n≤m
279                    for v=1:4
```

```
                    if PIXH(v,T(n))==1 && PIXgru(T(n)+NH(v))≠PIXgru(P1)
                    PIXgru(T(n)+NH(v))=PIXgru(P1);m=m+1;T(m)=T(n)+NH(v);
                    end
                  end
                  n=n+1;
                end
                else
                  T=P1; m=1; n=1;    % m: contador de T
               while n≤m
                for v=1:4
                 if PIXH(v,T(n))==1 && PIXgru(T(n)+NH(v))≠PIXgru(P2)
                 PIXgru(T(n)+NH(v))=PIXgru(P2);m=m+1;T(m)=T(n)+NH(v);
                 end
                end
                n=n+1;
               end
              end
              PIXH(E1,P1)=1; PIXH(E2,P2)=1; % esta conexión  se hace
              % al final para que la reagrupación no se afecte. Note
              % que este "if" no tiene un "else", pues en el caso de
              % que  PIXgru(P1) = PIXgru(P2)  entonces  dicha arista
              % formarÌa un ciclo, luego  NO  debe incluirse.
              end
          end
        end
      end
      end % fin del if PIXgf
      j=j+1;
    end  % fin del  while Jpar1
   end  % fin del if dIMIJ
  end % fin del contador I
 end  % fin del contador J
 %   PIXH  guarda la información del bosque soporte generado
%———————————— FIN DEL BOSQUE GENERADO ————————————————————————————
% ——————————————————————— Coloreado ———————————————————————————————
h=1; kk=0;                          % kk: contador de los grupos finales
while h≤Q
    if PIXcol(h)==−1
        PIXcol(h)=1;              % Coloreamos el pixel con 1.
                                  % En T guardaremos los pixeles del árbol.
        L=1;                      % L será el contador de los sucesivos
        T(L)=h;                   % elementos del árbol.
        K=1;  kk=kk+1;  PIXgf(h)=kk;
        while K≤L
            W=T(K);               % W guarda el número del nodo  T(K)
            for v=1:4
                Wi=W+NH(v);
                if PIXH(v,W)==1 && PIXcol(Wi)==−1 % para H1
                   L=L+1; T(L)=Wi; PIXcol(Wi)=1; % nodo visitado
                   DE=sqrt(sum((PIXdat(:,W)−PIXdat(:,Wi)).^2));
                   if DE < L1
                       PIXgf(Wi)=PIXgf(W);
                   else
                       kk=kk+1;
                      PIXgf(Wi)=kk; PIXH(NH1(v),Wi)=0; PIXH(v,W)=0;
                   end
                end
```

```
337             end
338             K=K+1;
339         end
340     end
341     h=h+1;
342 end
343 PIXgfA=PIXgf;
344 if za==1    % Primera intersección
345     PIXHOLD=PIXH;
346 end
347 %————————————————————— Intersección anterior ——————————————————————
348 SGRUPO=zeros(t1+1,kk); % guarda número de elementos (y suma) por grupo
349 MGRUP=zeros(r,s);
350 for i=1:r
351     for j=1:s
352         T=PIXgf((i-1)*s+j);
353         MGRUP(i,j)=T;
354         for k=1:t1
355             SGRUPO(k,T)=SGRUPO(k,T)+B(i,j,k);
356         end
357         SGRUPO(t1+1,T)=SGRUPO(t1+1,T)+1;
358     end
359 end
360 % cálculo de promedios
361 PROME=zeros(t1,kk);    %kk es el total de grupos
362 for k=1:t1
363     PROME(k,:)=SGRUPO(k,:)./SGRUPO(t1+1,:);
364 end
365 %————————————————————————————— Fin ——————————————————————————————
366
367 A1(:,:,za)=MGRUP;
368
369 for i=1:r
370     for j=1:s
371         for k=1:t1
372             AVERAG(i,j,k,za)=PROME(k,MGRUP(i,j));
373         end
374     end
375 end
376
377 PIXHOLD=PIXH; % guarda los enlaces para la siguiente intersección.
378 grupo(za)=kk; % número de grupos en cada alpha
379 za=za+1;
380 disp('salida de matrices i')
381 end   % fin del contador de los umbrales ALPHA
382 %————————————————————————————— Fin ——————————————————————————————
383
384 [A,B,C]=size(A1);
385 for G=1:C
386 QQ=contornoe(A1(:,:,G),A,B);
387 CT(:,:,G)=QQ;
388 end
389 toc
390
391 cform1=makecform('lab2srgb');
392 for G=1:C
393     filename=sprintf('imagenqqba%d.bmp',G); % genera nombres
```

```
394     imwrite(CT(:,:,G),filename); % guarda las imágenes (formato bmp)
395 end
396
397 if typeim==1  % Crea imágenes con los promedios a color (Lab a rgb)
398     for G=1:C
399     EDW(:,:,1)=AVERAG(:,:,1,G); % L
400     EDW(:,:,2)=AVERAG(:,:,2,G); % a
401     EDW(:,:,3)=AVERAG(:,:,3,G); % b
402     EDW1=applycform(EDW,cform1);
403     filename1=sprintf('imagenpromcolora%d.bmp',G);
404     imwrite(EDW1,filename1);
405     end
406 end
407
408 if typeim==0  %  Crea imágenes con promedios en grises
409     for G=1:C
410     filename1=sprintf('imagenpromgrayb%d.bmp',G);
411     imwrite(AVERAG(:,:,1,G),filename1);
412     end
413 end
```

A.3.2. contornoe.m

```
1  function CT=contornoe(GR,r,s)
2
3  % Esta función recibe una matriz de grupos GT y genera un contorno
4  % alrededor de cada grupo para visualizar el borde respectivo
5
6  %[r,s]=size(GR);
7  CT=zeros(2*r+1,2*s+1);
8
9  for i=1:(r-1)
10     for j=1:(s-1)
11         if GR(i,j)≠GR(i,j+1)
12             CT(2*i,2*j+1)=1; % punto de la forma  Pix - * - Pix
13         end
14         if GR(i,j)≠GR(i+1,j)
15             CT(2*i+1,2*j)=1; % punto de la forma  Pix / * / Pix
16         end
17         if GR(i,j)≠GR(i+1,j) || GR(i,j)≠GR(i,j+1) ||...
18            GR(i,j)≠GR(i+1,j+1) || GR(i,j+1)≠GR(i+1,j+1) ||...
19            GR(i+1,j)≠GR(i+1,j+1) || GR(i+1,j)≠GR(i,j+1)
20             CT(2*i+1,2*j+1)=1; % punto con pixeles en las diagonales
21         end
22     end
23 end
24
25 for L=1:(s-1)
26     if GR(r,L)≠GR(r,L+1)
27         CT(2*r,2*L+1)=1; % punto de la forma  Pix - * - Pix
28     end
29 end
30
31 for L=1:(r-1)
```

```
    if GR(L,s)≠GR(L+1,s)
        CT(2*L+1,2*s)=1; % punto de la forma  Pix / * / Pix
    end
end
```

A.3.3. hierarchicalf.m

```
clear all
clc

IMAG=input('image: ');

%————————————————————color————————————————————————
C=makecform('srgb2lab');
lab=applycform(IMAG,C);
Aredu=lab2double(lab);
%————————————————————————————————————————————————————
%————————————————Escala e grises ————————————————————
%Aredu=double(IMAG);   % gray-scale

d=1; %red: dxd
DELTI=100; % part I de la matriz
DELTJ=100; % part J de la matriz
TOL=2;     % Tolerancia de grupos fuertes
umbral=101;
%————————————————————————————————————————————————————
%———————————————— Mallado ———————————————————————————

[r,s,t]=size(Aredu);  S1=floor(s/d);  R1=floor(r/d);
d1=(r/d-R1)*d; d2=(s/d-S1)*d; % particion dominio

if d1>0
    R1=R1+1;
end
if d2>0
    S1=S1+1;
end
Baux=zeros(R1,S1,t);

for K=1:t
 for I=1:R1
    for J=1:S1
        suma=0;
        if I==R1 && d1>0
            da=d1;
        else
            da=d;
        end
        if J==S1 && d2>0
            db=d2;
        else
            db=d;
        end
        for k=1:da
```

```
            for l=1:db
                Ik=(I-1)*d+k;  Jl=(J-1)*d+l;
                suma=suma+Aredu(Ik,Jl,K);
            end
        end
        Baux(I,J,K)=suma/(da*db); % agregacion = average
    end  % fin de J
 end  % fin de I
end  % fin de K
Aredu1=Baux;  [r,s,t]=size(Aredu1);

%------------------------------ Fin mallado ------------------------------
%-------------------------------------------------------------------------

%-------------------------------- Umbrales -------------------------------
ALP=alfavalue(Aredu1,umbral,TOL);
w4=size(ALP,2);

SALIDA1=zeros(r,s,w4-1); SALIDA2=SALIDA1;
sf=s; rf=r;
%-------------------------------------------------------------------------
% ---------------- Particion de la matriz ------------------------------
PartI=floor(r/DELTI); PartJ=floor(s/DELTJ); extendI=0; extendJ=0;
if PartI<(r/DELTI)
    extendI=1;   % caso I final
end
if PartJ<(s/DELTJ)
    extendJ=1;   % caso J final
end
maxI=PartI+extendI; maxJ=PartJ+extendJ;
tic
DHP=[]; DVP=[]; MF=[]; w1F=[]; LLF=[]; DIMIJ=[];% Guarda resultados IJ
for I=1:maxI
     %auxI=0;
     I1=(I-1)*DELTI+1;
     if I<maxI
         I2=I1+DELTI-1;
     else
        I2=rf; %auxI=1;
     end
     for J=1:maxJ
        %auxJ=0;
        J1=(J-1)*DELTJ+1;
        if J<maxJ
            J2=J1+DELTJ-1;
        else
            J2=sf; %auxJ=1;
        end
        Aredu=Aredu1(I1:I2,J1:J2,:);
        [r,s,t]=size(Aredu);
        hierarchicalbase      % Este script realiza la segmentacion
                              % jerarquica usando Aredu, TOL y ALP
        SALIDA1(I1:I2,J1:J2,:)=B;
        SALIDA2(I1:I2,J1:J2,:)=B1;
     end
end
%-------------------------------------------------------------------------
```

```
%---------------------------------- salidas ----------------------------------------
for Q=1:w4-1
    figure, imshow(SALIDA1(:,:,Q));
end
%-----------------------------------------------------------------------------------
[r,s,t]=size(SALIDA1); B2=zeros(r,s);
for i=1:w4-1
suma=zeros(r,s);
for y=1:i
suma=suma+(1/i).*SALIDA1(:,:,y);
end
B2(:,:,i)=suma;
end
for Q=1:w4-1
    figure, imshow(B2(:,:,Q)); % B2 muestra imagenes con filtrado de
end                            % de ruido.
%-----------------------------------------------------------------------------------
%--------------------- guardar imagenes ------------------------------------------
for G=1:w4-1
     filename1=sprintf('aguila%d.bmp',G);
     imwrite(B(:,:,G),filename1);
end
```

A.3.4. alfavalue.m

```
function ALP=alfavalue(Aredu1,umbral,TOL)
% Funcion que crea un vector de umbrales ALF(1 x umbral) dada una
% matriz "Aredu1"

[r,s,t]=size(Aredu1);
%---------------------------- Red: T1,T2,T3,T4-------------------------------------
[DH,DV]=redT1(Aredu1,r,s);
%---------------------------- Matrices  Ai y  Aj ----------------------------------
Ai(1,:)=ones(1,s);
for i=2:r
    Ai(i,:)=Ai(i-1,:)+1;
end
Aj(:,1)=ones(r,1);
for j=2:s
    Aj(:,j)=Aj(:,j-1)+1;
end
DV1=DV'; Aiv=Ai(2:r,1:s)'; Ajv=Aj(2:r,1:s)'; uno=ones((r-1)*s,1);
H=cat(2,Aiv(:),Ajv(:),DV1(:),uno); % horizontal
DH1=DH'; Aih=Ai(1:r,2:s)'; Ajh=Aj(1:r,2:s)'; dos=2.*ones(r*(s-1),1);
V=cat(2,Aih(:),Ajh(:),DH1(:),dos); % vertical
D1=cat(1,H,V); % D1 guarda las aristas: D1=[i,j,d,(1 or 2)]
%-----------------------------------------------------------------------------------
%---------------------------- M=D1 ordenado ----------------------------------------
w1=size(D1,1);
[M1,IX]=sort(D1(:,3),'ascend');
M=D1;
for j=1:w1
    M(j,:)=D1(IX(j),:);
end
```

```
i=1; J=0; w1a=size(M,1);
while i≤w1a
    if M(i,3) ≤ TOL
        i=i+1;
    else
        J=i-1; i=w1a+2;
    end
end
A=M(J+1:end,:);

w3=size(A,1); ALP=[];
for i=1:umbral
    palfa=w3-4*((umbral-i)^2);  % distribucion de alphas
    ALP(i)=A(palfa,3);
end

end
```

A.3.5. redT1.m

```
function [DH,DV]=redT1(IMA,r1,s1)
% Funcion para crear la red T2 para la imagen IMA
% de dimension rxs en el espacio Lab.

DH1=IMA; DH1(:,1,:)=[]; DH2=IMA; DH2(:,s1,:)=[];
DH=sqrt(sum((abs(DH1-DH2)).^2,3));

DV1=IMA; DV1(1,:,:)=[]; DV2=IMA; DV2(r1,:,:)=[];
DV=sqrt(sum((abs(DV1-DV2)).^2,3));
end
```

A.3.6. redT2.m

```
function [DH,DV,DDA,DDD]=redT2(IMA,r1,s1)
% Funcion para crear la red T2 para la imagen IMA
% de dimension rxs en el espacio Lab.

DH1=IMA; DH1(:,1,:)=[]; DH2=IMA; DH2(:,s1,:)=[];
DH=sqrt(sum((abs(DH1-DH2)).^2,3));

DV1=IMA; DV1(1,:,:)=[]; DV2=IMA; DV2(r1,:,:)=[];
DV=sqrt(sum((abs(DV1-DV2)).^2,3));

DDA1=IMA; DDA1(1,:,:)=[]; DDA1(:,s1,:)=[];
DDA2=IMA; DDA2(:,1,:)=[]; DDA2(r1,:,:)=[];
DDA=sqrt(sum((abs(DDA1-DDA2)).^2,3));

DDD1=IMA; DDD1(1,:,:)=[]; DDD1(:,1,:)=[];
DDD2=IMA; DDD2(:,s1,:)=[]; DDD2(r1,:,:)=[];
DDD=sqrt(sum((abs(DDD1-DDD2)).^2,3));
end
```

A.3.7. redT3.m

```
function [DH,DV,DHL,DVL]=redT3(IMA,r,s)
% Funcion para crear la red T3 para la imagen IMA
% de dimension rxs en el espacio Lab.

DH1=IMA; DH1(:,1,:)=[]; DH2=IMA; DH2(:,s,:)=[];
DH=sqrt(sum((abs(DH1-DH2)).^2,3));

DV1=IMA; DV1(1,:,:)=[]; DV2=IMA; DV2(r,:,:)=[];
DV=sqrt(sum((abs(DV1-DV2)).^2,3));

DHL1=IMA; DHL1(:,1:2,:)=[]; DHL2=IMA; DHL2(:,s-1:s,:)=[];
DHL=sqrt(sum((abs(DHL1-DHL2)).^2,3));

DVL1=IMA; DVL1(1:2,:,:)=[]; DVL2=IMA; DVL2(r-1:r,:,:)=[];
DVL=sqrt(sum((abs(DVL1-DVL2)).^2,3));
end
```

A.3.8. redT4.m

```
function [DH,DV,DDA,DDD,DHL,DVL]=redT4(IMA,r,s)
% Funcion para crear la red T4 para la imagen IMA
% de dimension rxs en el espacio Lab.

DH1=IMA; DH1(:,1,:)=[]; DH2=IMA; DH2(:,s,:)=[];
DH=sqrt(sum((abs(DH1-DH2)).^2,3));

DV1=IMA; DV1(1,:,:)=[]; DV2=IMA; DV2(r,:,:)=[];
DV=sqrt(sum((abs(DV1-DV2)).^2,3));

DDA1=IMA; DDA1(1,:,:)=[]; DDA1(:,s,:)=[];
DDA2=IMA; DDA2(:,1,:)=[]; DDA2(r,:,:)=[];
DDA=sqrt(sum((abs(DDA1-DDA2)).^2,3));

DDD1=IMA; DDD1(1,:,:)=[]; DDD1(:,1,:)=[];
DDD2=IMA; DDD2(:,s,:)=[]; DDD2(r,:,:)=[];
DDD=sqrt(sum((abs(DDD1-DDD2)).^2,3));

DHL1=IMA; DHL1(:,1:2,:)=[]; DHL2=IMA; DHL2(:,s-1:s,:)=[];
DHL=sqrt(sum((abs(DHL1-DHL2)).^2,3));

DVL1=IMA; DVL1(1:2,:,:)=[]; DVL2=IMA; DVL2(r-1:r,:,:)=[];
DVL=sqrt(sum((abs(DVL1-DVL2)).^2,3));
end
```

A.3.9. hierarchicalbase.m

```
% Algoritmo que realiza una segmentacion jerarquica usando la tecnica
```

```
% de perfil (almacenamiento y busqueda vectorial), dado ALP, umbral
%------------------------- Net: T1,T2,T3,T4-----------------------------------
[DH,DV]=redT1(Aredu,r,s);
%-------------------------------------------------------------------------------
%------------------ Posicion de matrices Ai y Aj ---------------------------
Ai=[]; Aj=[];
Ai(1,:)=ones(1,s);
for i=2:r
    Ai(i,:)=Ai(i-1,:)+1;
end
Aj(:,1)=ones(r,1);
for j=2:s
    Aj(:,j)=Aj(:,j-1)+1;
end
DV1=DV'; Aiv=Ai(2:r,1:s)'; Ajv=Aj(2:r,1:s)'; uno=ones((r-1)*s,1);
H=cat(2,Aiv(:),Ajv(:),DV1(:),uno); % horizontal
DH1=DH'; Aih=Ai(1:r,2:s)'; Ajh=Aj(1:r,2:s)'; dos=2.*ones(r*(s-1),1);
V=cat(2,Aih(:),Ajh(:),DH1(:),dos); % vertical
D1=cat(1,H,V); % D1 guarda las aristas de: D1=[i,j,d,(1 or 2)]
%------------ AT identifica a cada nodo con un numero ---------------------
%-------------------------------------------------------------------------------
%-------------------------------M = D1 ordenado ------------------------------
w1=size(D1,1);
[M1,IX]=sort(D1(:,3),'ascend');
M=D1;
for j=1:w1
    M(j,:)=D1(IX(j),:);
end
i=1; J=0; w1a=size(M,1);
while i<=w1a
    if M(i,3) <= TOL
        i=i+1;
    else
        J=i-1; i=w1a+2;
    end
end
%MF=M(J+1:end,:);  %MF= M > alpha_tau
A=M(J+1:end,:);

DHa=DH<=TOL;  DVa=DV<=TOL;  grI=zeros(r,s);
H1=[zeros(r,1),DHa];  H2=[DHa,zeros(r,1)];
V1=[zeros(1,s);DVa];  V2=[DVa;zeros(1,s)];
kk=0;
for j=1:s
    for i=1:r
        if grI(i,j)==0 % nodo no visitado
            kk=kk+1; grI(i,j)=kk;
            T=[i,j]; m=1; n=1;
            while n<=m
                I=T(n,1); J=T(n,2);
                if H2(I,J)==1 && grI(I,J+1)==0
                    grI(I,J+1)=kk;
                    m=m+1; T(m,1)=I; T(m,2)=J+1; %T(m,:)=[I,J+1];
                end
                if V1(I,J)==1 && grI(I-1,J)==0
                    grI(I-1,J)=kk;
                    m=m+1; T(m,1)=I-1; T(m,2)=J; %T(m,:)=[I-1,J];
```

```
                end
                if H1(I,J)==1 && grI(I,J-1)==0
                    grI(I,J-1)=kk;
                    m=m+1; T(m,1)=I; T(m,2)=J-1; %T(m,:)=[I,J-1];
                end
                if V2(I,J)==1 && grI(I+1,J)==0
                    grI(I+1,J)=kk;
                    m=m+1; T(m,1)=I+1; T(m,2)=J; %T(m,:)=[I,J+1];
                end
                n=n+1;
            end % fin del 'while'
        end % fin del 'if'
    end
end
MGF=grI;
GRaux=ones(1,kk);  GR=cumsum(GRaux);
w3=size(A,1);   w4=size(ALP,2);
i=1;
for j=1:w4
    UMBR=ALP(j); ii=1; LL(j)=0;
    if i<1
        i=1;
    end
    while i≤w3 && ii==1
        if A(i,3)<UMBR
            i=i+1;
        else
            i=i-1; ii=0;
            if i>0
                LL(j)=i;
            else
                LL(j)=1;
            end
        end
    end
    if LL(j)==0
        LL(j)=w3;
    end
end
%-----------------------------------Fin LL ---------------------------------
%--------------------------------------------------------------------------
%------------------------ Proceso jerarquizado ----------------------------
MGFold=ones(r,s);     % agrupacion inicial
Mo=GR;  grupos=zeros(r,s);
for j=w4:-1:2
    ALPj=ALP(j); LLj=LL(j);
    %-MM: edge-matrix created by using the criterium--
    M1=A(1:LLj,:); % size(M1)=(LL,4).
    w1=size(A,1);
    if LLj<w1
       M2=A(w1:-1:LLj+1,:);
    else
      M2=[];
    end
    MM=cat(1,M2,M1);    % size(M2)=(w1-LLj,4).
    [MGFold,EMo,EGR1,ES]=cluster1(LLj,w1,MM,MGF,MGFold,Mo,GR,kk);
    grupos(:,:,w4-j+1)=MGFold;
```

```
A=M1;      % nuevo "A" para el ALPj siguiente
end
%-------------------------------------------------------------------------
CGR=zeros(1,kk,w4-1);  % conteo
SGR=zeros(t,kk,w4-1);  % suma
for q=1:w4-1
   for i=1:r
      for j=1:s
        CGR(1,grupos(i,j,q),q)=CGR(1,grupos(i,j,q),q)+1;
        for k=1:t
           SGR(t,grupos(i,j,q),q)=SGR(t,grupos(i,j,q),q)+Aredu(i,j,t);
        end
     end
   end
end
%-------------------------------------------------------------------------
%-----------------------------------BORDE---------------------------------

B=zeros(r,s,w4-1); % B saves  segmentations
for q=1:w4-1
    B(:,:,q)=BORDE(grupos(:,:,q),CGR(:,:,q),r,s);
end
%-------------------------------------------------------------------------
%-------------------------Borde filtrado----------------------------------
B1=B;
for q=1:w4-1
    for i=3:r-2
        for j=3:s-2
            if B1(i,j,q)==1
                W=B1(i-1:i+1,j-1:j+1,q); % 3x3
                Ta=sum(W(:));
                if Ta-B1(i,j,q)==0
                    B1(i,j,q)=0;
                else
                    W1=B1(i-2:i+2,j-2:j+2,q); % 5x5
                    Tb=sum(W1(:));
                    if Tb-Ta==0
                        B1(i-1:i+1,j-1:j+1,q)=zeros(3,3);
                    end
                end
            end
        end
    end
end
%-------------------------------------------------------------------------
%------------------------- Muestra resultados ----------------------------
for Q=1:w4-1
    figure, imshow(B(:,:,Q));
end
%-------------------------------------------------------------------------
%-------------------------guardar imagenes--------------------------------
for G=1:w4-1
     filename1=sprintf('aguila%d.bmp',G);
     imwrite(B(:,:,G),filename1);
end
```

A.3.10. cluster1.m

```
function [MGFold,EMo,EGR1,ES]=cluster1(LLj,w1,MM,MGF,MGFold,Mo,GR,kk)
% Algoritmo auxiliar para realizar  una segmentacion jerarquica usando
% la tecnica de "perfil"

S=[0,cumsum(ones(1,kk))]; % suma de grupos iniciales
GR1=GR;  MGF1=MGF;
EMo=Mo; EGR1=GR1; ES=S; % grupos finales luego de la particion
%-------------------------------------------------------------------------
L=1; % contador para la matriz MM
while L≤w1
    Ia=MM(L,1); Ja=MM(L,2); de=MM(L,3); tao=MM(L,4);
    if tao==1
        Ib=Ia-1; Jb=Ja; % v-aristas.
    else
        Ib=Ia; Jb=Ja-1; % h-aristas.
    end
    ga=MGF1(Ia,Ja); gb=MGF1(Ib,Jb);
    ka=GR1(ga); kb=GR1(gb); % grupos de (Ia,Ja) y (Ib,Jb).
    if ka==kb
        L=L+1; % no es usa esta arista pues forma un ciclo.
    else
        if MGFold(Ia,Ja)≠MGFold(Ib,Jb); % grupos diferentes anteriores
            L=L+1; % no se usa la arista para garantizar jerarquizacion
        else
         % en este caso: GR1(ga)=¬GR1(gb) & MGFold(Ia,Ja)=MGFold(Ib,Jb)
         % ---------------- adicionar la arista al arbol ------------------
            na=S(ka+1)-S(ka); nb=S(kb+1)-S(kb);
            Moka=Mo(S(ka)+1:S(ka+1)); Mokb=Mo(S(kb)+1:S(kb+1));
            if ka < kb
                Mo1=Mo(1:S(ka)); Mo2=Mo(S(ka+1)+1:S(kb));
                Mo3=Mo(S(kb+1)+1:end);
                if na ≤ nb % move ka to kb
                    Mo=[Mo1,Mo2,Mokb,Moka,Mo3];
                    GR1(Moka)=kb; % actualizar los grupos
                    S(ka+1:kb)=S(ka+1:kb)-na; % actualizar la suma
                else   % mover kb a ka
                    Mo=[Mo1,Moka,Mokb,Mo2,Mo3];
                    GR1(Mokb)=ka; % actualizar los grupos
                    S(ka+1:kb)=S(ka+1:kb)+nb; % actualizar la suma
                end
            else  % dual
                Mo1=Mo(1:S(kb)); Mo2=Mo(S(kb+1)+1:S(ka));
                Mo3=Mo(S(ka+1)+1:end);
                if na ≤ nb  % move ka to kb
                    Mo=[Mo1,Mokb,Moka,Mo2,Mo3];
                    GR1(Moka)=kb; % actualiza grupos
                    S(kb+1:ka)=S(kb+1:ka)+na; % actualiza suma
                else  % move kb to ka
                    Mo=[Mo1,Mo2,Moka,Mokb,Mo3];
                    GR1(Mokb)=ka; %refresh actualiza grupos
                    S(kb+1:ka)=S(kb+1:ka)-nb; % actualiza suma
                end
            end  % fin de 'if' ka < kb
```

```
54              %------ proceso auxiliar para unir aristas ------------------
55             if L>w1-LLj
56              ka=EGR1(ga); kb=EGR1(gb);
57              na=ES(ka+1)-ES(ka); nb=ES(kb+1)-ES(kb);
58              Moka=EMo(ES(ka)+1:ES(ka+1)); Mokb=EMo(ES(kb)+1:ES(kb+1));
59              if ka < kb
60                  Mo1=EMo(1:ES(ka)); Mo2=EMo(ES(ka+1)+1:ES(kb));
61                  Mo3=EMo(ES(kb+1)+1:end);
62                  if na ≤ nb
63                      EMo=[Mo1,Mo2,Mokb,Moka,Mo3];
64                      EGR1(Moka)=kb;
65                      ES(ka+1:kb)=ES(ka+1:kb)-na;
66                  else
67                      EMo=[Mo1,Moka,Mokb,Mo2,Mo3];
68                      EGR1(Mokb)=ka;
69                      ES(ka+1:kb)=ES(ka+1:kb)+nb;
70                  end
71              else  % dual
72                  Mo1=EMo(1:ES(kb)); Mo2=EMo(ES(kb+1)+1:ES(ka));
73                  Mo3=EMo(ES(ka+1)+1:end);
74                  if na ≤ nb
75                      EMo=[Mo1,Mokb,Moka,Mo2,Mo3];
76                      EGR1(Moka)=kb;
77                      ES(kb+1:ka)=ES(kb+1:ka)+na;
78                  else  % move kb to ka
79                      EMo=[Mo1,Mo2,Moka,Mokb,Mo3];
80                      EGR1(Mokb)=ka;
81                      ES(kb+1:ka)=ES(kb+1:ka)-nb;
82                  end
83              end
84             end
85              L=L+1;
86          end
87      end
88  end % fin del while 'L'
89
90  MGFold=EGR1(MGF1); % guarda los grupos para el siguiente paso
91  end
```

A.3.11. BORDE.m

```
1   function B=BORDE(G,C,r,s)
2   % Funcion para crear el borde de una particion conservando la dimension
3   % rxsvde la imagen original donde:
4   % G: matriz de grupos de (rxs)
5   % C: vector de conteo de grupos (1xkk)
6   % B: matriz de salida con los bordes (rxs)
7   B=zeros(r,s);
8   %n=20;
9   % horizontal
10  for i=1:r
11      for j=2:s
12          P=G(i,j); Q=G(i,j-1);
13          if P≠Q
```

```
            if C(P) ≤ C(Q)
                B(i,j)=1;
            else
                B(i,j−1)=1;
            end

        end
    end
end
% vertical
for i=2:r
    for j=1:s
        P=G(i,j); Q=G(i−1,j);
        if P≠Q
            if C(P) ≤ C(Q)
                B(i,j)=1;
            else
                B(i−1,j)=1;
            end

        end
    end
end

end
```

A.3.12. hierarchicalf8.m

```
% Genera una segmentacion jerarquica para una imagen con mallado de
% 8 vecinos.
clear all
clc

IMAG=input('image: ');
d=input('mallado: ');
DELTI=input('particion en I: ');
DELTJ=input('particion en J: ');
TOL=input('tolerancia: ');
umbral=input('numero de umbrales: ');
%————————————————————————————————————————————————————————————————
%———————————————————— color ————————————————————————————————————
C=makecform('srgb2lab');
lab=applycform(IMAG,C);
Aredu=lab2double(lab);
%————————————————————escala de grises————————————————————————————
% Aredu=double(IMAG);
%————————————————————————————————————————————————————————————————
[r,s,t]=size(Aredu);  S1=floor(s/d);  R1=floor(r/d);
d1=(r/d−R1)*d; d2=(s/d−S1)*d; % particion del dominio

if d1>0
    R1=R1+1;
end
if d2>0
```

```
    S1=S1+1;
end
Baux=zeros(R1,S1,t);

for K=1:t
 for I=1:R1
    for J=1:S1
        suma=0;
        if I==R1 && d1>0
            da=d1;
        else
            da=d;
        end
        if J==S1 && d2>0
            db=d2;
        else
            db=d;
        end
        for k=1:da
            for l=1:db
                Ik=(I-1)*d+k;  Jl=(J-1)*d+l;
                suma=suma+Aredu(Ik,Jl,K);
            end
        end
        Baux(I,J,K)=suma/(da*db); % agregador = average
    end  % of J
 end  % of I
end  % of K
Aredu1=Baux;
[r,s,t]=size(Aredu1);

%------------------------------- END mallado ------------------------------
%--------------------------------------------------------------------------
%------------------------------- ALFA valores -----------------------------
ALP=alfavalue8(Aredu1,umbral,TOL);
w4=size(ALP,2); SALIDA1=zeros(r,s,w4-1); SALIDA2=SALIDA1; sf=s; rf=r;
%--------------------------------------------------------------------------
% ----------------- Particion de la matriz ------------------------------
PartI=floor(r/DELTI); PartJ=floor(s/DELTJ); extendI=0; extendJ=0;
if PartI<(r/DELTI)
    extendI=1;   % caso I final
end
if PartJ<(s/DELTJ)
    extendJ=1;   % caso J final
end
maxI=PartI+extendI; maxJ=PartJ+extendJ;
DHP=[]; DVP=[]; MF=[]; w1F=[]; LLF=[]; DIMIJ=[];% save each IJ results
for I=1:maxI
     I1=(I-1)*DELTI+1;
     if I<maxI
         I2=I1+DELTI-1;
     else
        I2=rf;
     end
     for J=1:maxJ
        J1=(J-1)*DELTJ+1;
        if J<maxJ
```

```
            J2=J1+DELTJ-1;
        else
            J2=sf;
        end
        Aredu=Aredu1(I1:I2,J1:J2,:); [r,s,t]=size(Aredu);
        hierarchicalbase8  % segmentacion jerarquica con 8 vecinos
        SALIDA1(I1:I2,J1:J2,:)=B;  SALIDA2(I1:I2,J1:J2,:)=B1;
     end
end
%-------------------------------------------------------------------------
for Q=1:w4-1
    figure, imshow(SALIDA1(:,:,Q));
end
%-------------------------------------------------------------------------
[r,s,t]=size(SALIDA1); B2=zeros(r,s);
for i=1:w4-1
suma=zeros(r,s);
for y=1:i
suma=suma+(1/i).*SALIDA1(:,:,y);
end
B2(:,:,i)=suma;
end
for Q=1:w4-1
   figure, imshow(B2(:,:,Q));
end
%-------------------------------------------------------------------------
%------------------------- guardar imagen --------------------------------
for G=1:w4-1
    filename1=sprintf('aguila%d.bmp',G);
    imwrite(B(:,:,G),filename1);
end
```

A.3.13. alfavalue8.m

```
function ALP=alfavalue8(Aredu1,umbral,TOL)
% Funcion que crea un vector de umbrales ALF(1 x umbral) dada una
% matriz "Aredu1" para 8 vecinos

[r,s,t]=size(Aredu1);
%-------------------------------------------------------------------------
%------------------------- Red: T1,T2,T3,T4------------------------------
[DH,DV,DDA,DDD]=redT2(Aredu1,r,s);
%-------------------------------------------------------------------------
% ------------------------ Matrices Ai y Aj-------------------------------
Ai(1,:)=ones(1,s);
for i=2:r
    Ai(i,:)=Ai(i-1,:)+1;
end
Aj(:,1)=ones(r,1);
for j=2:s
    Aj(:,j)=Aj(:,j-1)+1;
end
DV1=DV'; Aiv=Ai(2:r,1:s)'; Ajv=Aj(2:r,1:s)'; uno=ones((r-1)*s,1);
H=cat(2,Aiv(:),Ajv(:),DV1(:),uno); % horizontal
```

```
DH1=DH'; Aih=Ai(1:r,2:s)'; Ajh=Aj(1:r,2:s)'; dos=2.*ones(r*(s-1),1);
V=cat(2,Aih(:),Ajh(:),DH1(:),dos); % vertical
%---------------
Dia1=DDA'; Aid=Ai(2:r,1:s-1)'; Ajd=Aj(2:r,1:s-1)';
tres=3.*ones((r-1)*(s-1),1);
Diaa=cat(2,Aid(:),Ajd(:),Dia1(:),tres); % diagonal derecha

Dia2=DDD'; Aidd=Ai(2:r,2:s)'; Ajdd=Aj(2:r,2:s)';
cuatro=4.*ones((r-1)*(s-1),1);
Diab=cat(2,Aidd(:),Ajdd(:),Dia2(:),cuatro); % diagonal izquierda

D1=cat(1,H,V,Diaa,Diab);% D1 guarda las aristas: D1=[i,j,d,(1 or 2)]
%-------------------------------------------------------------------------
%------------------------- M=D1 ordenado ---------------------------------
w1=size(D1,1); [M1,IX]=sort(D1(:,3),'ascend');  M=D1;
for j=1:w1
    M(j,:)=D1(IX(j),:);
end
i=1; J=0; w1a=size(M,1);
while i<=w1a
    if M(i,3) <= TOL
        i=i+1;
    else
        J=i-1; i=w1a+2;
    end
end
A=M(J+1:end,:);
%------------------------------ Umbrales ---------------------------------
w3=size(A,1); ALP=[];
for i=1:umbral
    palfa=w3-5*((umbral-i)^2);  % distribucion de umbrales
    ALP(i)=A(palfa,3);
end

end
```

A.3.14. hierarchicalbase8.m

```
% Algoritmo que realiza una segmentacion jerarquica usando la tecnica
% de perfil (almacenamiento y busqueda vectorial), dado ALP, umbral
% y con 8 vecinos.
%------------------------------ Net: T1,T2,T3,T4-------------------------
[DH,DV,DDA,DDD]=redT2(Aredu,r,s);
%-------------------------------------------------------------------------
%------------------------- Matrices Ai y Aj ------------------------------
Ai=[]; Aj=[];
Ai(1,:)=ones(1,s);
for i=2:r
    Ai(i,:)=Ai(i-1,:)+1;
end
Aj(:,1)=ones(r,1);
for j=2:s
    Aj(:,j)=Aj(:,j-1)+1;
end
```

```
%AT=(Ai-1).*s+Aj;
DV1=DV'; Aiv=Ai(2:r,1:s)'; Ajv=Aj(2:r,1:s)'; uno=ones((r-1)*s,1);
H=cat(2,Aiv(:),Ajv(:),DV1(:),uno); % horizontal
DH1=DH'; Aih=Ai(1:r,2:s)'; Ajh=Aj(1:r,2:s)'; dos=2.*ones(r*(s-1),1);
V=cat(2,Aih(:),Ajh(:),DH1(:),dos); % vertical

Dia1=DDA'; Aid=Ai(2:r,1:s-1)'; Ajd=Aj(2:r,1:s-1)';
tres=3.*ones((r-1)*(s-1),1);
Diaa=cat(2,Aid(:),Ajd(:),Dia1(:),tres); % diagonal derecha

Dia2=DDD'; Aidd=Ai(2:r,2:s)'; Ajdd=Aj(2:r,2:s)';
cuatro=4.*ones((r-1)*(s-1),1);
Diab=cat(2,Aidd(:),Ajdd(:),Dia2(:),cuatro); % diagonal izquierda

D1=cat(1,H,V,Diaa,Diab); %D1 guarda las aristas: D1=[i,j,d,(1 or 2)]
%-------------------------- M=D1 ordenado ---------------------------------
w1=size(D1,1);  [M1,IX]=sort(D1(:,3),'ascend');  M=D1;
for j=1:w1
    M(j,:)=D1(IX(j),:);
end
i=1; J=0; w1a=size(M,1);
while i<=w1a
    if M(i,3) <= TOL
        i=i+1;
    else
        J=i-1; i=w1a+2;
    end
end
A=M(J+1:end,:);
%--------------------------------------------------------------------------

DHa=DH<=TOL;  DVa=DV<=TOL;  DDAa=DDA<=TOL;  DDDa=DDD<=TOL;

grI=zeros(r,s);
H1=[zeros(r,1),DHa];  H2=[DHa,zeros(r,1)];
V1=[zeros(1,s);DVa];  V2=[DVa;zeros(1,s)];

DDA1=[zeros(1,s);[DDA,zeros(r-1,1)]];
DDA2=[[zeros(r-1,1),DDA];zeros(1,s)];
DDD1=[zeros(1,s);[zeros(r-1,1),DDD]];
DDD2=[[DDD,zeros(r-1,1)];zeros(1,s)];

kk=0;
for j=1:s
    for i=1:r
        if grI(i,j)==0    % nodo no visitado
            kk=kk+1; grI(i,j)=kk;
            T=[i,j]; m=1; n=1;
            while n<=m
                I=T(n,1); J=T(n,2);
                if H2(I,J)==1 && grI(I,J+1)==0
                    grI(I,J+1)=kk;
                    m=m+1; T(m,1)=I; T(m,2)=J+1; %T(m,:)=[I,J+1];
                end
                if V1(I,J)==1 && grI(I-1,J)==0
                    grI(I-1,J)=kk;
```

```
                    m=m+1; T(m,1)=I-1; T(m,2)=J; %T(m,:)=[I-1,J];
                end
                if H1(I,J)==1 && grI(I,J-1)==0
                    grI(I,J-1)=kk;
                    m=m+1; T(m,1)=I; T(m,2)=J-1; %T(m,:)=[I,J-1];
                end
                if V2(I,J)==1 && grI(I+1,J)==0
                    grI(I+1,J)=kk;
                    m=m+1; T(m,1)=I+1; T(m,2)=J; %T(m,:)=[I,J+1];
                end
                %-------------------------------
                if DDA1(I,J)==1 && grI(I-1,J+1)==0
                    grI(I-1,J+1)=kk;
                    m=m+1; T(m,1)=I-1; T(m,2)=J+1; %T(m,:)=[I-1,J+1];
                end
                if DDA2(I,J)==1 && grI(I+1,J-1)==0
                    grI(I+1,J-1)=kk;
                    m=m+1; T(m,1)=I+1; T(m,2)=J-1; %T(m,:)=[I+1,J-1];
                end
                if DDD1(I,J)==1 && grI(I-1,J-1)==0
                    grI(I-1,J-1)=kk;
                    m=m+1; T(m,1)=I-1; T(m,2)=J-1; %T(m,:)=[I-1,J-1];
                end
                if DDD2(I,J)==1 && grI(I+1,J+1)==0
                    grI(I+1,J+1)=kk;
                    m=m+1; T(m,1)=I+1; T(m,2)=J+1; %T(m,:)=[I+1,J+1];
                end
                n=n+1;
            end % fin del 'while'
        end % fin del 'if'
    end
end
MGF=grI;
GRaux=ones(1,kk);  GR=cumsum(GRaux);

w3=size(A,1);  w4=size(ALP,2);  i=1;
for j=1:w4
    UMBR=ALP(j); ii=1; LL(j)=0;
    if i<1
        i=1;
    end
    while i≤w3 && ii==1
        if A(i,3)<UMBR
            i=i+1;
        else
            i=i-1; ii=0;
            if i>0
                LL(j)=i;
            else
                LL(j)=1;
            end
        end
    end
    if LL(j)==0
        LL(j)=w3;
    end
end
```

```
%-------------------------------------------------------------------------
%---------------------- Proceso jerarquizado -----------------------------
MGFold=ones(r,s); % grupos iniciales
Mo=GR;   grupos=zeros(r,s);
for j=w4-1:2
    ALPj=ALP(j); LLj=LL(j);
    %---------MM: matriz con las aristas segun el griterio usado--------
    M1=A(1:LLj,:); % size(M1)=(LL,4).
    w1=size(A,1);
    if LLj<w1
       M2=A(w1-1:LLj+1,:);
    else
      M2=[];
    end
    MM=cat(1,M2,M1);    % size(M2)=(w1-LLj,4).
    [MGFold,EMo,EGR1,ES]=cluster88(LLj,w1,MM,MGF,MGFold,Mo,GR,kk);
    grupos(:,:,w4-j+1)=MGFold;
    A=M1; % nuevo A para el siguiente ALPj
end
%-------------------------------------------------------------------------
CGR=zeros(1,kk,w4-1);  % conteo
SGR=zeros(t,kk,w4-1);  % suma
for q=1:w4-1
   for i=1:r
      for j=1:s
        CGR(1,grupos(i,j,q),q)=CGR(1,grupos(i,j,q),q)+1;
        for k=1:t
           SGR(t,grupos(i,j,q),q)=SGR(t,grupos(i,j,q),q)+Aredu(i,j,t);
        end
      end
   end
end
%-------------------------------------------------------------------------
%-------------------------------BORDE-------------------------------------
B=zeros(r,s,w4-1); % B guarda la secuencia
for q=1:w4-1
    B(:,:,q)=BORDE(grupos(:,:,q),CGR(:,:,q),r,s);
end
%-------------------------------------------------------------------------
%---------------------------Border filtrado-------------------------------
B1=B;
for q=1:w4-1
    for i=3:r-2
        for j=3:s-2
            if B1(i,j,q)==1
                W=B1(i-1:i+1,j-1:j+1,q); % 3x3
                Ta=sum(W(:));
                if Ta-B1(i,j,q)==0
                    B1(i,j,q)=0;
                else
                    W1=B1(i-2:i+2,j-2:j+2,q); % 5x5
                    Tb=sum(W1(:));
                    if Tb-Ta==0
                        B1(i-1:i+1,j-1:j+1,q)=zeros(3,3);
                    end
                end
            end
```

```
188         end
189     end
190 end
191 %-------------------------------------------------------------------------
192 %-------------------------------muestra imagenes--------------------------
193 for Q=1:w4-1
194      figure, imshow(B(:,:,Q));
195 end
196 %-------------------------------------------------------------------------
197 %-------------------------------guarda imagenes---------------------------
198 for G=1:w4-1
199     filename1=sprintf('aguila%d.bmp',G);
200     imwrite(B(:,:,G),filename1);
201 end
```

A.3.15. cluster88.m

```
1  function [MGFold,EMo,EGR1,ES]=cluster88(LLj,w1,MM,MGF,MGFold,Mo,GR,kk)
2  % Algoritmo auxiliar para realizar  una segmentacion jerarquica usando
3  % la tecnica de "perfil" para 8 vecinos
4
5  S=[0,cumsum(ones(1,kk))]; % suma inicial
6  GR1=GR;  MGF1=MGF;
7  EMo=Mo; EGR1=GR1; ES=S; % grupos finales luego de la particion
8  %-------------------------------------------------------------------------
9  L=1; % contador para MM
10 while L≤w1
11     Ia=MM(L,1); Ja=MM(L,2); de=MM(L,3); tao=MM(L,4);
12     if tao==1
13         Ib=Ia-1; Jb=Ja; % v-aristas.
14     else
15         if tao==2
16             Ib=Ia; Jb=Ja-1; % h-aristas.
17         else
18             if tao==3
19                 Ib=Ia-1; Jb=Ja+1;
20             else % tao=4
21                 Ib=Ia-1; Jb=Ja-1;
22             end
23         end
24     end
25     ga=MGF1(Ia,Ja); gb=MGF1(Ib,Jb);
26     ka=GR1(ga); kb=GR1(gb); % grupos de (Ia,Ja) y (Ib,Jb).
27     if ka==kb
28         L=L+1; % no es usa esta arista pues forma un ciclo.
29     else
30         if MGFold(Ia,Ja)≠MGFold(Ib,Jb); % grupos diferentes anteriores
31             L=L+1; % no se usa la arista para garantizar jerarquizacion
32         else
33             % en este caso: GR1(ga)=¬GR1(gb) & MGFold(Ia,Ja)=MGFold(Ib,Jb)
34             %--------------- adicionar la arista al arbol ----------------
35             na=S(ka+1)-S(ka); nb=S(kb+1)-S(kb);
36             Moka=Mo(S(ka)+1:S(ka+1)); Mokb=Mo(S(kb)+1:S(kb+1));
37             if ka < kb
```

```
38                  Mo1=Mo(1:S(ka)); Mo2=Mo(S(ka+1)+1:S(kb));
39                  Mo3=Mo(S(kb+1)+1:end);
40                  if na ≤ nb % move ka to kb
41                      Mo=[Mo1,Mo2,Mokb,Moka,Mo3];
42                      GR1(Moka)=kb; % actualizar los grupos
43                      S(ka+1:kb)=S(ka+1:kb)-na; % actualizar la suma
44                  else   % move kb to ka
45                      Mo=[Mo1,Moka,Mokb,Mo2,Mo3];
46                      GR1(Mokb)=ka; % actualizar los grupos
47                      S(ka+1:kb)=S(ka+1:kb)+nb; % actualizar la suma
48                  end
49              else  % dual
50                  Mo1=Mo(1:S(kb)); Mo2=Mo(S(kb+1)+1:S(ka));
51                  Mo3=Mo(S(ka+1)+1:end);
52                  if na ≤ nb  % move ka to kb
53                      Mo=[Mo1,Mokb,Moka,Mo2,Mo3];
54                      GR1(Moka)=kb; % actualizar los grupos
55                      S(kb+1:ka)=S(kb+1:ka)+na; % actualizar la suma
56                  else  % move kb to ka
57                      Mo=[Mo1,Mo2,Moka,Mokb,Mo3];
58                      GR1(Mokb)=ka; % actualizar los grupos
59                      S(kb+1:ka)=S(kb+1:ka)-nb; % actualizar la suma
60                  end
61              end  % fin de 'if' ka < kb
62              %----- proceso auxiliar para unir aristas ------------------
63             if L>w1-LLj
64              ka=EGR1(ga); kb=EGR1(gb);
65              na=ES(ka+1)-ES(ka); nb=ES(kb+1)-ES(kb);
66              Moka=EMo(ES(ka)+1:ES(ka+1)); Mokb=EMo(ES(kb)+1:ES(kb+1));
67              if ka < kb
68                  Mo1=EMo(1:ES(ka)); Mo2=EMo(ES(ka+1)+1:ES(kb));
69                  Mo3=EMo(ES(kb+1)+1:end);
70                  if na ≤ nb % move ka to kb
71                      EMo=[Mo1,Mo2,Mokb,Moka,Mo3];
72                      EGR1(Moka)=kb;
73                      ES(ka+1:kb)=ES(ka+1:kb)-na;
74                  else
75                      EMo=[Mo1,Moka,Mokb,Mo2,Mo3];
76                      EGR1(Mokb)=ka;
77                      ES(ka+1:kb)=ES(ka+1:kb)+nb;
78                  end
79              else  % dual
80                  Mo1=EMo(1:ES(kb)); Mo2=EMo(ES(kb+1)+1:ES(ka));
81                  Mo3=EMo(ES(ka+1)+1:end);
82                  if na ≤ nb  % move ka to kb
83                      EMo=[Mo1,Mokb,Moka,Mo2,Mo3];
84                      EGR1(Moka)=kb;
85                      ES(kb+1:ka)=ES(kb+1:ka)+na;
86                  else  % move kb to ka
87                      EMo=[Mo1,Mo2,Moka,Mokb,Mo3];
88                      EGR1(Mokb)=ka;
89                      ES(kb+1:ka)=ES(kb+1:ka)-nb;
90                  end
91              end  % fin de 'if' ka < kb
92             end
93              L=L+1;
94          end
```

```
95      end
96  end % fin de while 'L'
97
98  MGFold=EGR1(MGF1); % guarda los grupos para el siguiente paso
99  end
```

Bibliografía

[1] R.K. Ahuja, Magnanti, T. L., and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliks, USA, 1993.

[2] A. Amo, J. Montero, G. Biging, and V. Cutello. Fuzzy classification systems. *European Journal of Operational Research*, 156(2):495–507, 2004.

[3] S. Andrews, G. Hamarneh, and A. Saad. Fast random walker with priors using precomputation for interactive medical image segmentation. *Lecture Notes in Computer Science*, 6363:9–16, 2010.

[4] G. J. Babu and E. D. Feigelson. *Statistical Challenges in Modern Astronomy II*. Springer, New York, 1996.

[5] E. M. L. Beale. Euclidean cluster analysis. *Bulletin of the International Statistical Institute: Proceedings of the 37th Session*, 2:92–94, 1969.

[6] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech.: Theory Exp.*, page 10008, 2008.

[7] Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. *Proceedings of International Conference on Computer Vision*, 1:105–112, 2001.

[8] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[9] U. Brandes, D. Delling, M. Gaertler, R. Goerke, M. Hoefer, Z. Nikoloski, , and D. Wagner. On modularity clustering. *EEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, 2008.

[10] U. Brandes and T. Erlebach. *Network Analysis: Methodological Foundations*. Springer, 2005.

[11] H. Bustince, E. Barrenechea, and M. Pagola. Image thresholding using restricted equivalence functions and maximizing the measures of similarity. *Fuzzy Sets and Systems*, 158(5):496–516, 2007.

[12] H. Bustince, E. Barrenechea, and M. Pagola. Generation of interval-valued fuzzy and atanassov's intuitionistic fuzzy connectives from fuzzy connectives and from k? operators: Laws for conjunctions and disjunctions, amplitude. *International Journal of Intelligent Systems*, 23(6):680–714, 2008.

[13] H. Bustince, E. Barrenechea, M. Pagola, and J. Fernandez. Interval-valued fuzzy sets constructed from matrices: Application to edge detection. *Fuzzy Sets and Systems*, 160(13):1819–1840, 2009.

[14] H. Bustince, V. Mohedano, E. Barrenechea, and M. Pagola. Definition and construction of fuzzy di-subsethood measures. *Information Sciences*, 176(21):3190–3231, 2006.

[15] H. Bustince, M. Pagola, and E. Barrenechea. Construction of fuzzy indices from fuzzy di-subsethood measures: Application to the global comparison of images. *Information Sciences*, 177(3):906–929, 2007.

[16] H. Bustince, M. Pagola, E. Barrenechea, J. Fernandez, P. Melo Pinto, P. Couto, H. R. Tizhoosh, and J. Montero. Ignorance functions. an application to the calculation of the threshold in prostate ultrasound images. *Fuzzy Sets and Systems*, 161(1):20–36, 2010.

[17] G. Celeux and G. Govaert. A classification em algorithm for clustering and two stochastic versions. *Computational Statistics and Data Analysis*, (4):315–332, 1992.

[18] C. Chakrapani. *Statistics in Market Research*. Arnold, London, 2004.

[19] C. Christoudias, B. Georgescu, and P. Meer. Synergism in low-level vision. *Proceedings of the 16th International Conference on Pattern Recognition, Proceedings of the 16th International Conference on Pattern Recognition, Quebec City, Canada*, 4:150–155, 2002.

[20] K. Chung, W. Yang, and W. Yan. Efficient edge-preserving algorithm for color contrast enhancement with application to color image segmentation. *Journal of Visual Communication and Image Representation*, 19(5):299–310, 2008.

[21] C. Çigla and A. A. Alatan. Efficient graph-based image segmentation via speeded-up turbo pixels. *17th IEEE International Conference on Image Processing*, pages 3013–3016, 2010.

[22] A. Clauset, C. Moore, and M.E.J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453:98–101, 2008.

[23] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70:066111, 2004.

[24] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[25] J. D. Cortese. The array of today: biomolecule arrays become the 21st century test tube. *The Scientist*, 15:25, 2000.

[26] L. Doneti and M. A. Muñoz. Improved spectral algorithm for the detection of network communities. *AIP Conf. Proc. 779. arXchic:Physics/0504059.*, 104, 2005.

[27] D. Dubios and H. Prade. Ranking fuzzy numbers in the setting of possibility theory. *Information Sciences*, 30:183–224, 1983.

[28] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, New York, 1973.

[29] M. B. Eisen, P. T. Spellman, P. O. Brown, and P. O. Botstein. Cluster analysis and cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences of the United States of America*, 95:14863–14868, 1998.

[30] M. Espinilla, J. Montero, and J. T. Rodriguez. Computational intelligence in decision making. *International Journal of Computational Intelligence Systems*, (in press).

[31] M. Ester, R. Ge, B. J. Gao, Z. Hu, and Ben-Moshe. Joint cluster analysis of attribute data and relationship data: the connected k-center problem. *ACM Transactions on Knowledge Discovery from Data*, 2(2), 2008.

[32] E. Estrada and N. Hatano. Communicability in complex networks. *Physical Review E*, 77:036111, 2008.

[33] E. Estrada and N. Hatano. Communicability graph and community structures in complex networks. *Applied Mathematics and Computation*, 214:500–511, 2009.

[34] G. Facchinetti and R. G. Ricci. A characterization of a general class of ranking functions on triangular fuzzy numbers. *Fuzzy Set and Systems*, 146:297–312, 2004.

[35] M. Faúndez-Abans, M. I. Ormeno, and M. de Oliveira-Abans. Classification of planetary nebulae by cluster analysis and artificial neural networks. *Astronomy Astrophys- Astronomy Astrophysics Supplement*, 116:395–402, 1996.

[36] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[37] S. Fortunato. Community detection in graphs. *Physical Reports*, 486(75174), 2010.

[38] F. Fouss and J.M. Renders. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. on Knowl. and Data Eng.*, 19:35–369, 2007.

[39] L. C. Freeman. *The Development of Social Network Analysis: A Study in the Sociology of Science.* Empirical Press, 2004.

[40] H. P. Friedman and J. Rubin. On some invariant criteria for grouping data. *Journal of the American Statistical Association*, 62:1159–1178, 1967.

[41] K. S. Fu and J. K. Mui. A survey on image segmentation. *Pattern Recognition*, 13:3–16, 1981.

[42] A. Pradera G. Beliakov and T. Calvo. *Aggregation functions: a guide for practitioners*. Springer, 2008.

[43] M. Girvan and M.E.J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12):7821–7826, 2002.

[44] D. Gomez, J. Figueira, and A. Eusebio. Modeling centrality measures in social network analysis using bi-criteria network flow optimization problems. *European Journal of Operational Research*, 2013.

[45] D. Gómez and J. Montero. Fuzzy sets in remote sensing classification. *Soft Computing*, 12:243–249, 2008.

[46] D. Gómez, J. Montero, and G. Biging. Accuracy statistics for judging soft classification. *International Journal of Remote Sensing*, 29:693–709, 2008.

[47] D. Gómez, J. Montero, and G. Biging. Improvements to remote sensing using fuzzy classification, graphs and accuracy statistics. *Pure and Applied Geophysics*, 165(1555-1575), 2008.

[48] D. Gómez, J. Montero, and J. Yáñez. A coloring algorithm for image classification. *Information Sciences*, 176(3645-3657), 2006.

[49] D. Gómez, J. Montero, J. Yáñez, and C. Poidomani. A graph coloring algorithm approach for image segmentation. *Omega*, 35:173–183, 2007.

[50] D. Gómez and J. Montero, J. y Yáñez. A coloring fuzzy approach for image clasification. *Information Science*, 2006.

[51] M. Gondran and M. Minoux. *Graphs and Algorithms*. John Wiley and Sons, 1984.

[52] R. González, R. Woods, and S. Eddins. *Digital image processing using MATLAB*. MacGraw Hill, 2010.

[53] A. D. Gordon. *Cluster validation, in Data Science, Classification and Related Methods*. Springer-Verlag, 1998.

[54] L. Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006.

[55] L. Grady and G. Funka-Lea. Multi-label image segmentation for medical applications based on graph-theoretic electrical potentials. *Proceedings of the 8th ECCV Workshop on Computer Vision Approaches to Medical Image Analysis and Mathematical Methods in Biomedical Image Analysis*, pages 230–245, 2004.

[56] L. Grady and E. L. Schwartz. Isoperimetric graph partitioning for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):469–475, 2006.

[57] L. Grady and A. K. Sinop. Fast approximate random walker segmentation using eigenvector precomputation. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[58] V. Grau, A. U. J. Mewes, M. Alcaniz, R. Kikinis, and S. K. Warfield. Improved watershed transform for medical image segmentation using prior information. *IEEE Transactions on Medical Imaging*, 23(4):447–458, 2004.

[59] P. E. Green, R. E. Frank, and P. J. Robinson. Cluster analysis in test market selection. *Management Science*, 13:387–400, 1967.

[60] S Gregory. Fuzzy overlapping communities in networks. *Journal of Statistical Mechanics: Theory and Experiment Volume*, doi:10.1088/1742-5468/2011/02/P02017, 2011.

[61] W. Groissboeck, E. Lughofer, and S. Thumfart. Associating visual textures with human perceptions using genetic algorithms. *Information Sciences*, 180:2065–2084, 2010.

[62] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2009.

[63] D. S. Hochbaum. Polynomial time algorithms for ratio regions and a variant of normalized cut. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):889–898, 2010.

[64] M.-H. Hsieh and C.L. Magee. A new method for finding hierarchical subgroups from networks. *Social Networks*, 32:234–244, 2010.

[65] IEEE Signal Processing Society Press. *A review of recent evaluation methods for image segmentation*, Kuala Lumpur, Malaysia, 2001. Proccedings of the VI International Symposium on Signal Processing and Its Applications.

[66] O. Kabva and S. Seikkala. On fuzzy metric spaces. *Fuzzy Set and Systems,*, pages 215–229, 1984.

[67] J. Kleinberg. Authoritative sources in a hyperlinked environment. *In Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[68] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: A comparative analysis. *Phys. Rev. E*, 80:056117, 2009.

[69] P. Langfelder, B. Zhang, and S. Horvath. Defining clusters from a hierarchical cluster ree; the dynamic tree cut package for r. *Bioinformatics*, 24:719–720, 2008.

[70] D. Lusseau, K. Schneider, O. Boisseau, P. Haase, E. Slooten, and S.M. Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. can geographic isolation the bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. can geographic isolation explain this unique trait? *Behavioral Ecology and Sociobiology*, 54:396–405, 2003.

[71] M. Madhubanti and C. Amitava. A hybrid cooperative–comprehensive learning based pso algorithm for image segmentation using multilevel thresholding. *Expert Systems with Applications*, 34(2):1341–1350, 2008.

[72] C. Martin, T. Barlow, and W. Barnhart L. et al. The galaxy evolution explorer. *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 336–350, 2003.

[73] M. Mignotte. Segmentation by fusion of histogram-based k-means clusters in different color spaces. *IEEE Transactions on Image Processing*, 17(5):780–787, 2008.

[74] G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50:159–179, 1985.

[75] J. Montero, D. Gómez, and H. Bustince. On the relevance of some families of fuzzy sets. *Fuzzy Sets and Systems*, 158:2429–2442, 2007.

[76] J. Montero and L. Martínez. Upgrading ideas about the concept of soft computing. *International Journal of Computational Intelligence Systems*, 3:144–147, 2010.

[77] M. E. J. Newman. Communities, modules and large-scale structure in networks. *Nature Physics*, 8:25–31, 2012.

[78] M.E.J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69:066133, 2004.

[79] M.E.J. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27(1):39–54, 2005.

[80] M.E.J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, 2004.

[81] N. Ohta and A. Robertson. *Colorimetry. Fundamentals and Applications*. John Wiley and Sons, 2005.

[82] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[83] P. M. Pardalos, T. Mavridov, and J. Xue. *The graph coloring problem: A bibliographic survey. Hand.book of Combinatorial Optimization*. Kluber Academic Publishers, 1998.

[84] E. S. Paykel. Classification of depressed patients: a cluster analysis derived grouping. *British Journal of Psychiatry*, 118:275–288, 1979.

[85] B. Peng and O. Veksler. Parameter selection for graph cut based image segmentation. *British Machine Vision Conference*, 2008.

[86] Z. Petera, V. Boussone, C. Bergote, and F. Peyrina. A constrained region growing approach based on watershed for the segmentation of low contrast structures in bone micro-ct images. *Pattern Recognition*, 41:2358–2368, 2008.

[87] I. Pilowsky, S. Levine, and D. M. Boulton. The classification of depression by numerical the classification of depression by numerical taxonomy. *British Journal of Psychiatry*, 115:937–945, 1969.

[88] P. Pons and M. Lapaty. Computing communities in large networks using random walks. *http://arxiv.org/abs/physics/0512106*, 2005.

[89] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101:2658–2663, 2004.

[90] S. R. Rao, H. Mobahi, A. Y. Yang, S. S. Sastry, and Y. Ma. Natural image segmentation with adaptive texture and boundary encoding. *9th Asian Conference on Computer Vision*, 2009.

[91] L. G. Roberts. *Machine perception of three-dimensional solids.* Cambridge. MA: MIT Press, 1965.

[92] M. Saerens, F. Fouss, L. Yen, and P. Dupont. The principal components analysis of a graph, and its relationships to spectral clustering. In *ECML 2004, Lecture Notes in Artificial Intelligence*, 2004.

[93] J. P. Scott. *Social Network Analysis.* Thousand Oaks, CA: SAGE, 2nd edition, 2000.

[94] S. Selinski and K. Ickstadt. Cluster analysis of genetic and epidemiological data in cluster analysis of genetic and epidemiological data in molecular epidemiology. *Journal of Toxicology and Environmental Health*, 71:835–844, 2008.

[95] N. Senthilkumaran and R. Rajesh. Edge detection techniques for image segmentation a survey of soft computing approaches. *International Journal of Recent Trends in Engineering*, 1(2):250–254, 2009.

[96] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2001.

[97] F. Tavares-Pereira, J.R. Figueira, V. Mousseau, and B Roy. Multiple criteria districting problems: The public transportation network pricing system of the paris region. *Annals of Operations Research*, 154:69–92, 2007.

[98] J. R. Tyler, D. M. Wilkinson, and B. A. Huberman. Email as spectroscopy: Automated discovery of community structure within organizations. *Communities and technologies (Kluwer, B.V., Deventer, The Netherlands)*, pages 81–96, 2003.

[99] G. Ugarriza, L. Saber, and E. Vantaram et al. Automatic image segmentation by dynamic region growth and multiresolution merging. *IEEE Transactions on Image Processing*, 18(10):2275–2288, 2009.

[100] Stijn van Dongen. A cluster algorithm for graphs. *Technical Report INS-R0010, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam*, 2000.

[101] C. R. Wallace and D. M. Boulton. An information measure for classification. *Computer Computer Journal*, 11:185–194, 1968.

[102] J. Wang, L. Ju, and X. Wang. An edge-weighted centroidal voronoi tessellation model for image segmentation. *IEEE Transactions on Image Processing*, 18(8):1844–1858, 2009.

[103] X. Wang and E. E. Kerre. Reasonable properties for the ordering of fuzzy quantities i. *Fuzzy Set and Systems*, 118:375–385, 2001.

[104] Xiang-Yang Wang and Bu. Juan. A fast and robust image segmentation using fcm with spatial information. *Digital Signal Processing*, 20(4):1173–1182, 2010.

[105] J. H. Ward. Hierarchical groupings to optimize an objective function. *Journal of the American Statistical Association*, (58):236–244, 1963.

[106] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences)*. Cambridge University Press, Cambridge, U.K., 1 edition, 1994.

[107] B. Wellman and S. Berkowitz. *Social Structures: A Network Approach*. Cambridge University Press, 1988.

[108] D. M. Wilkinson and B. A. Huberman. A method for finding communities of related genes. *Proc. Natl. Acad. Sci. U.S.A*, 101:5241–5248, 2004.

[109] D. M. Witten and R. Tibshirani. Supervised multidimensional scaling for visualization, classification and bipartite ranking. *Computational Statistics and Data Analysis*, 55:789–801, 2010.

[110] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, 1993.

[111] L. Xu, W. Li, and D. Schuurmans. Fast normalized cut with linear constraints. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2866–2873, 2009.

[112] J. Yáñez, S. Muñoz, and J. Montero. Graph coloring graph coloring inconsistencies in image segmentation. *Computer Engineering and Information Sciences*, 1:435–440, 2008.

[113] W. Yang, J. Cai, J. Zheng, and J. Luo. User-friendly interactive image segmentation through unified combinatorial user inputs. *IEEE Transactions on Image Processing*, 19(9):2470–2479, 2010.

[114] L. Yen, F. Fouss, C Decaestecker, and M. Saerens. Graph nodes clustering based on the commute-time kernel. *Lecture notes in Computer Science*, LNAI(4426):1037–1045, 2007.

[115] Z.D. Yu, O.C. Au, and R.B. Zou. An adaptive unsupervised approach toward pixel clustering and color image segmentation. *Pattern Recognition*, 43(5):1889–1906, 2010.

[116] M.E. Yuksel and M. Borlu. Accurate segmentation of dermoscopic images by image thresholding based on type-2 fuzzy logic. *IEEE Transactions on Fuzzy Systems*, 17(4):976–982, 2010.

[117] N. Yuruk, M. Mete, Xu X., and T.A.J. Schweiger. A divisive hierarchical structural clustering algorithm for networks. In *Proceedings of the 7th IEEE International Conference on Data Mining Workshops*, pages 441–448, 2007.

[118] W.W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.

[119] L. A. Zadeh. Fuzzy sets. *Information and Control,*, 8:338–353, 1965.

[120] J. Zhang, J. Zheng, and J. Cai. A diffusion approach to seeded image segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2125–2132, 2010.

[121] Y. J. Zhang. *Image Segmentation*. Science Plublisher, Beijing. China, 2001.

[122] Y. J. Zhang. *Advances in image an video segmentation*. Hershey. PA:IRM Press, 2006.

Summary

In this work we introduce a hierarchical clustering algorithm in networks based upon a first divisive stage to break the graph and a second linking stage which is used to join nodes. As a particular case, this algorithm is applied to a specific community detection problem in Social Networks, where a *betweenness* measure is considered for the divisive criterion a similarity measure associated to data is used for the linking criterion. Results are compared with some standard methodologies for hierarchical clustering problems, by considering some well-known examples as the *Karate Zachary Club* network, the *Dolphins* network, *Les Miserables* network and the *Authors Centrality* network. We also have applied the proposed algorithm to image segmentation.

Clustering pursues the arrangement of a family of items into homogeneous groups, taking into account inherent quantitative and qualitative information about them. However, the practical implementation of some clustering techniques requires certain mathematical assumptions that sometimes are difficult, if not impossible, to be checked. Some of these assumptions are quite often simply hidden in the interpreter's mind. Statistical independence, for example, represents a standard hypothetical reference, and many theoretical arguments apply only if such independence holds.

There is still a lot of work ahead on clustering models acknowledging how real objects are connected (see, e.g., [2] for an example of a clustering model without structure, and the relevant modification proposed in [75]). In fact, many sets of real data can be structured by means of a graph, i.e., a set of nodes linked by edges. If these edges are valued, we have a network.

Moreover, the relations between the endpoints of the edges could be so complex that a 1-dimensional view of reality cannot be accurate enough; our graph model should allow a more complex view, adding the representation of different aspects that explain reality.

The analysis of complex networks is a wide topic that attracts a growing interest in recent years. In particular, the study of the topology and properties of networks is playing an important role in computer science, biology and social sciences, among other fields (see for example [37, 80, 78, 106, 43]). In particular, complex social networks represent an interesting field of application. Discovering inherent communities and structures in a social network must be a main objective when we pursue a better understanding of a given network.

As in classical statistical clustering, networks can be analyzed using either a hierarchical or a non-hierarchical clustering approach. The main aim of non-hierarchical clustering methods, which are the most used in the literature, is to obtain a partition of the graph according to some objective criterion, producing a partition of the network

as output (see, for example, [37, 97] for various approaches and applications).

Nevertheless, in some situations, as for example in the splitting process of a company or a group of friends, this process is done in a dynamic way, so it would be relevant to know the whole splitting process instead of the final picture. In such cases, a hierarchical clustering approach seems more appropriate, since the overall splitting process can be visualized, bringing specific advantages. For example, as pointed out in [22], the knowledge of hierarchical structure can be used to predict missing connections in a more realistic way. For this reason, in this work, we have focused on hierarchical clustering networks. As stressed in [22] *hierarchy is a central organizing principle of complex networks, capable of offering insight into many network phenomena* (see also [30], where the importance of *understanding* a problem is stressed against automatic approaches to complex decision making problems).

Community detection problems (see, for example, [37, 97]) are usually defined as a clustering network problem, in which the main aim is to find a *good* partition of the network. In this paper we address a more complex problem, the *hierarchical clustering network problem* (HCNP), that pursues a *good* hierarchical partition of networks. This new problem gives more emphasis in the dynamic process of the clustering, rather than the final *picture* of the clustering process. We consider that quite often in social network analysis, hierarchical partitions are more informative than a single partition, once they can shows the evolution of how the groups are formed (agglomerative) or split (divisive), from the initial situation until the last step. As pointed out in [22], *hierarchy is a central organizing principle of complex networks, capable of offering insight into many network phenomena*. A hierarchical analysis of a network represents a much more informative output than a single non-hierarchical output.

Given the HCNP, in this paper we have proposed a new algorithm for networks that allows to cluster a set of related items, by means of a valued graph. Our Divide-and-Link algorithm amalgamates ideas from hierarchical agglomerative and divisive methods since it is based on the construction of a spanning forest, based on two complementary approaches to break the nodes in a first stage, and to join the nodes in a second stage. The core of the proposed algorithm is an iterative binary procedure for a graph, based upon the basic procedure already introduced by the authors in previous papers [48, 49]. Such a basic procedure was then introduced to obtain a segmentation of a digital image (see also [47, 45]), modeled as a particular network of pixels. In the new algorithm proposed here, however, there are no constraints on the structure of the network, allowing the clustering of arbitrary networks.

Another major achievement of the new proposed algorithm is its polynomial complexity. This fact, together with the proposed automatic dendrogram representation, our algorithm opens the possibility of a visual analysis of very large networks. Let us stress that visual representations are an absolute need when we are facing complex problems (see, e.g., [61]) and information is subject to some underlying structure (see also [30]).

Finally, in this paper we have adapted our Divide-and-Link algorithms, D&L and D&L fast, to deal with community detection problems, where the only available information is the graph of the network. For this application we have used two criteria: the betweenness measure, and a similarity measure for adjacent nodes defined in [60]. We have compared our algorithm in some well-known test examples in community detection, comparing it with the most effective hierarchical algorithms. The proposed algorithms have shown very good performance in terms of modularity.

To conclude, we would like to stress with other authors (see, e.g., [46]) the absolute need for alternative methodologies to assess the performance of clustering and classification algorithms for structured data. In the framework of algorithms for the clustering of networks we can find (see [37] for a general review) some measures to assess the quality of any particular partition of a graph. In particular, the concept of modularity introduced in [43] plays a key role in our approach. But existing measures have not been designed for the performance assessment of a hierarchical algorithm, in which the final output use to be a dendrogram, not a single partition. There are too many real situations in which the relations between objects are given in an imprecise way, especially in the field of social relations (see, e.g., [48]). Representing relations between objects by means of a valued graph has allowed us a interesting approach that fit into these problems without excessive computational complexity. How to extend the ideas and the algorithm her presented into a more general soft-computing framework is an important objective to be explored in the near future (see. e.g., [76] for a general discussion of the concept of soft computing).

In image segmentation, one of the main problems when working on theories involving segmentation thresholds is to choose a set of appropriate thresholds for the segmentation process. If we want to provide the threshold set explicitly, we need to manipulate the pixel values in any color space, and the distances between them in a measure space. However, in our proposed algorithm is more important the number of cutting edges (edges with large values) included in a given step of segmentation, that the threshold value α_i. Because of this, the only thing that matters to hierarchical segmentation proposed is the number of steps (the K value), we can even say how many cutting edges are to be incorporated in every step.

In the processing of the images of Figures 4.8 and 4.11, the CPU time was no more than 1 minute for each hierarchical image segmentation. The algorithm was run on a 1.7GHz Intel Core i5 with 3GB of RAM. The five thresholds have been chosen automatically by the algorithm in order to include a fixed size of cut edges ($100\,t^2$ cut edges are included on each step, for $t = 1, \ldots, 5$).

Finally, our proposed algorithm can be executed using coarse nets, which reduces the computational time substantially. The theory of aggregation functions can be used for this purpose, i.e., to relate sets of pixels through some aggregation function, so that results can be obtained more quickly and preserving a good classification. Performances of different aggregator functions in netwokrs can be a future research work.